

SONDERHEFT 53
Markt&Technik
ÖS 120,–/Sfr. 16,–/Lit. 16000
hfl. 21,–/dkr. 75,–/fmk 43,–
DM 16,–
DISKETTE IM HEFT
64'er
DAS BESTE
AUS 5 JAHREN
Spannend
Heiße Spiele für Joystickakrobaten
Neu zusammengestellt
10 Top-Programme aus allen Bereichen
Kaum zu glauben
PC-Simulation auf dem C 64
DISKETTE IM HEFT

# POWER-GAMES

Jedes Paket nur DM 49,–* (sFr 45,–/*öS 490,–*)

## ACTION • SPANNUNG • ABENTEUER

**Das Schwert Skar**
Skar verleiht seinem Träger elementare Kräfte. Es macht ihn unbesiegbar und unsterblich. Aber es ist gut versteckt! Wer es finden will, muß den Gefahren eines langen Weges trotzen.
**Bestell-Nr. 38784**

**Die Flucht der Sumpfgeister**
Als die Menschen begannen, die Sümpfe trocken zu legen, haben die Sumpfgeister mit dem einzigen vorhandenen magischen Staubsauger die Flucht zu einem weit entfernten Planeten ergriffen. Ein Sumpfgeist hat jedoch die Abreise verschlafen...
**Bestell-Nr. 38785**

**POWER-GAMES erhalten Sie im guten Fachhandel**

**Operation Ushkurat**
Sie sind mit einem Raumschiff unterwegs zu Friedensverhandlungen. Bei einer Reparatur wird die gesamte Mannschaft entführt...
**Bestell-Nr. 38765**

**Dungeon**
»Dungeon« ist eine Variante des legendären Spieleklassikers »PacMan«. Die Spielfigur bewegt sich durch ein Labyrinth. Eingebaute Türen, Teleporter sowie diverse Hilfsmittel helfen Ihnen, Geistern und Monstern aus dem Weg zu gehen...
**Bestell-Nr. 38760**

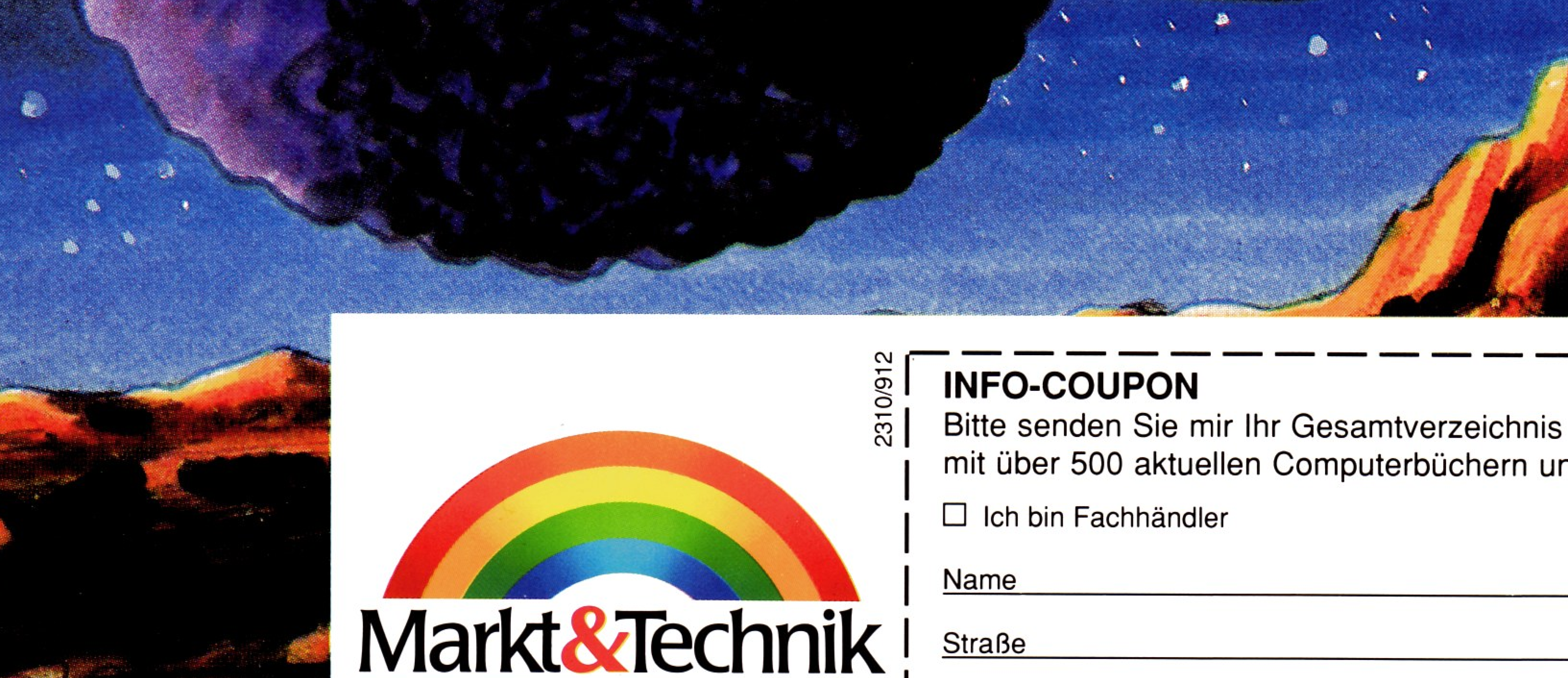
**Operation Feuersturm**
Sie sind »Mister James Bond« und haben 48 Stunden Zeit, eine gestohlene Atombombe zu finden – falls nicht, wird sie abgefeuert.
**Bestell-Nr. 38739**

**Howard the Coder**
Howard hat eine Spielidee. Leider stiehlt man seinen Computer und er sucht sich in einer Lagerhalle neue Hardware zusammen. Dabei muß er Hindernisse überwinden...
**Bestell.-Nr. 38705**

**Mit Jeans und Hellebarde**
Bei der Reparatur eines Schuppens stürzt die Decke herab und macht Sie kampfunfähig. Als Sie zu sich kommen entdecken Sie ein altes Buch mit merkwürdigen Buchstabenkombinationen. Sie wissen noch nicht, daß Sie Ihre Welt bereits verlassen haben...
**Bestell-Nr. 38718**

**Nippon – das ultimative Rollenspiel für C 64/C 128**
Toshiro begann, die zufällig entdeckten Schriftrollen zu lesen. Sie sahen abgegriffen und uralt aus... Vor Ihnen liegt ein Abenteuer, wie Sie es bisher nicht gekannt haben.
**Bestell-Nr. 38729**

* Unverbindliche Preisempfehlung.



Markt&Technik
Zeitschriften · Bücher
Software · Schulung

2310/912

**INFO-COUPON**
Bitte senden Sie mir Ihr Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software.

☐ Ich bin Fachhändler

Name ____________________

Straße ____________________

PLZ/Ort ____________________

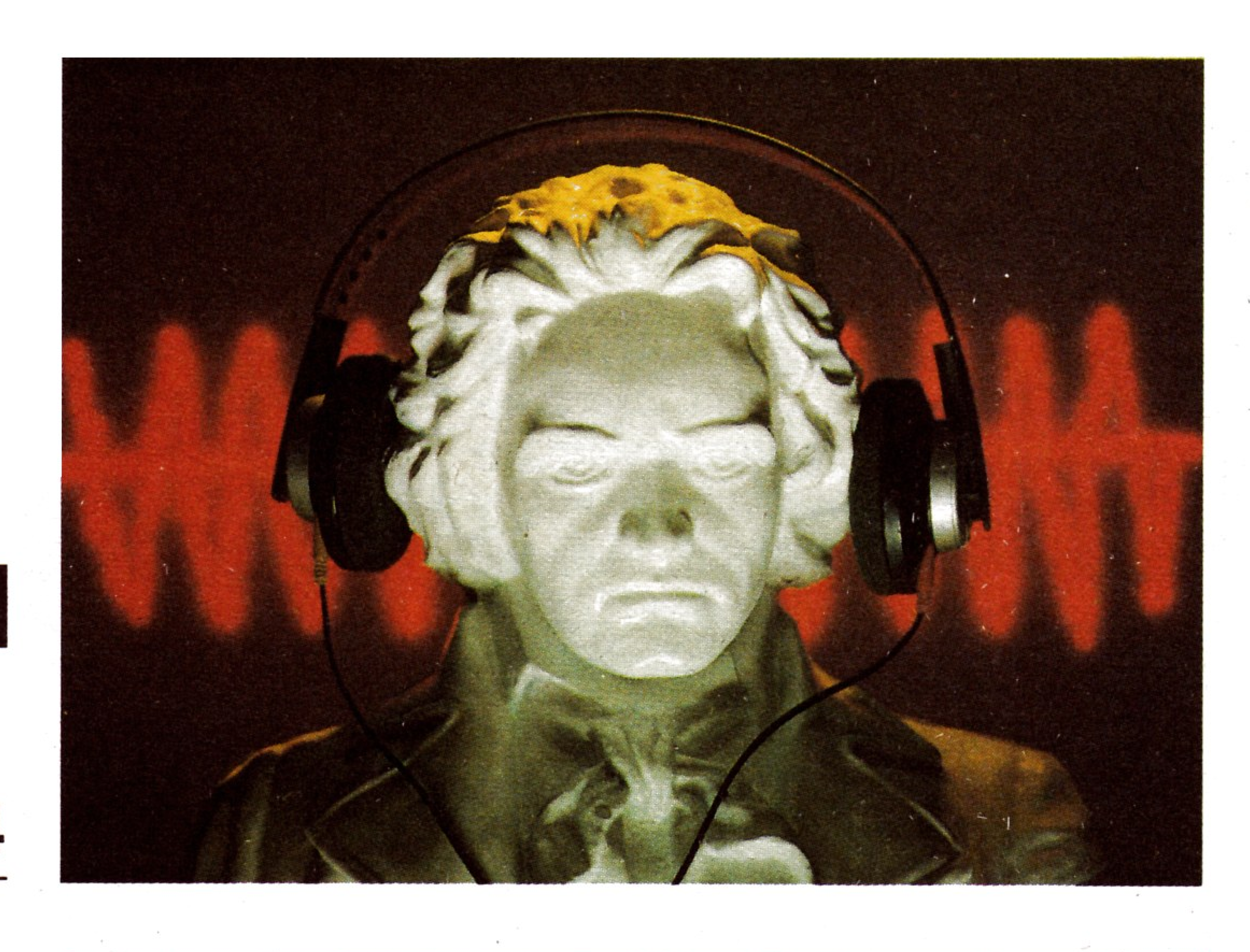
Bitte ausschneiden und senden an: Markt&Technik Verlag AG, Buch- und Software-Verlag, Hans-Pinsel-Str. 2, 8013 Haar bei München 64 SH 53



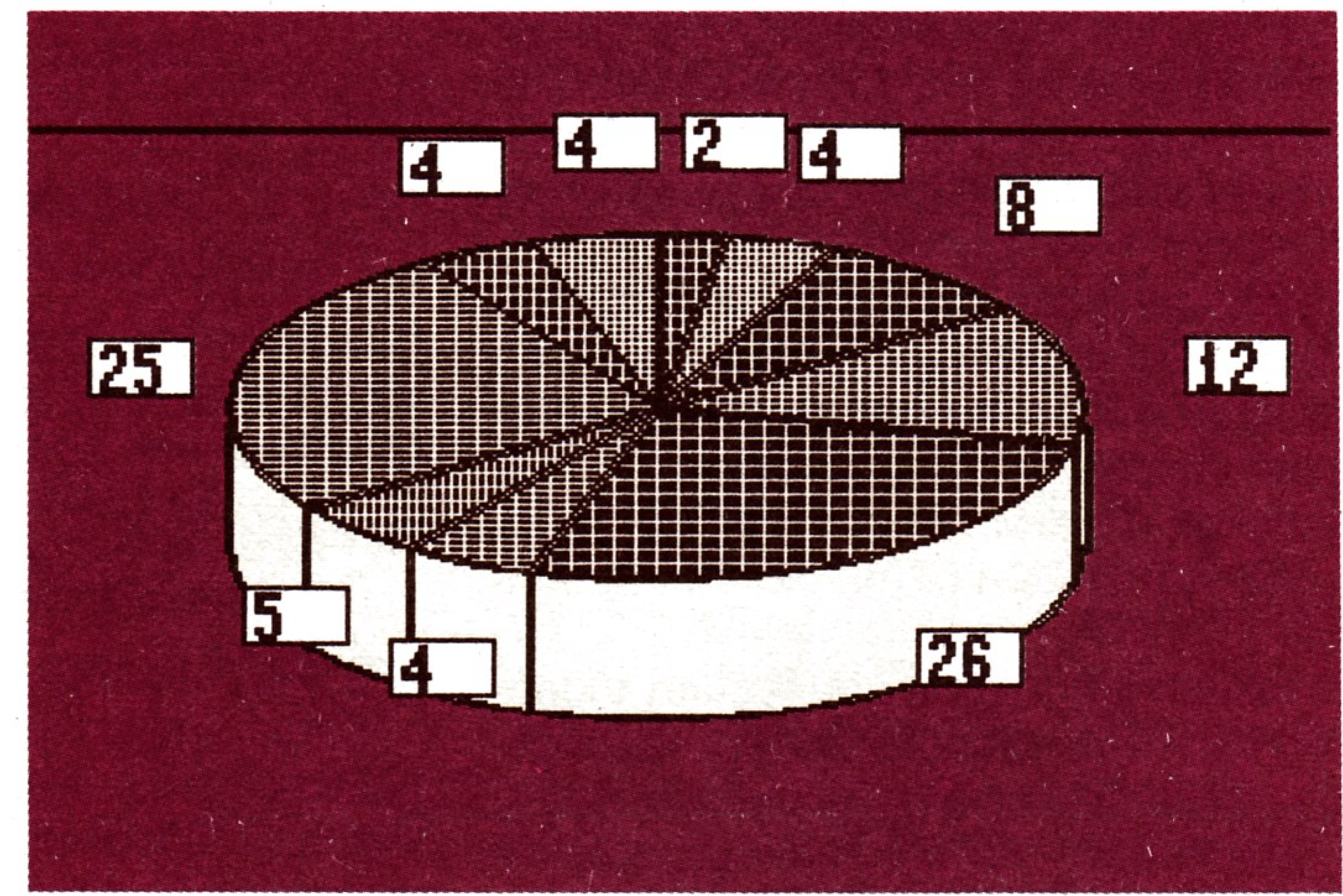




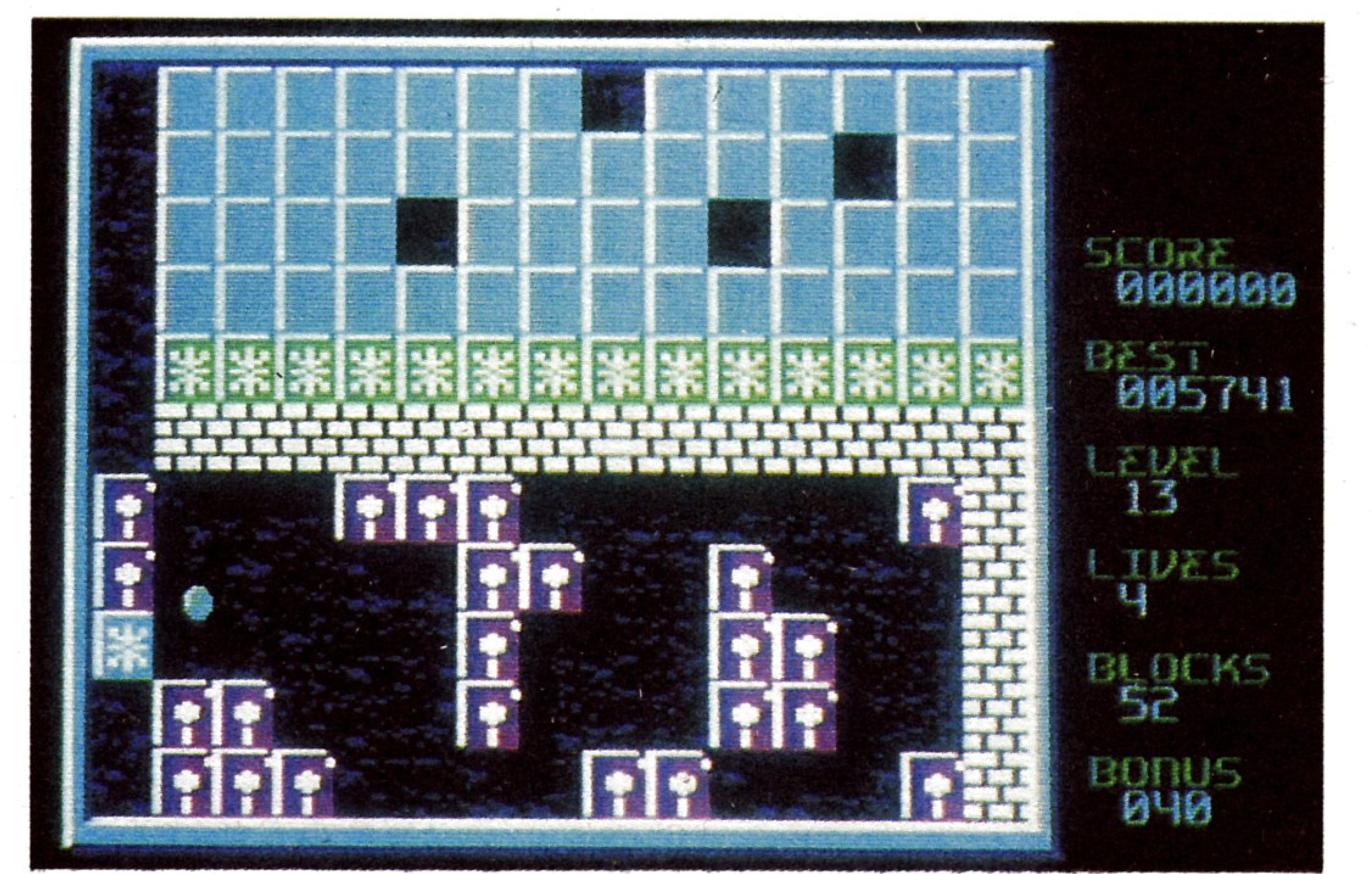




**Beethoven hätte seine Freude gehabt mit dem »Sound-Monitor«. Es ist kaum zu glauben, welche Musikstücke sich mit diesem Programm entwickeln lassen.**
**Seite 42**




**Solche Bilder lassen sich mit »Grafik 2001« schnell programmieren. Das Programm ist erstaunlich flott beim Erstellen der Grafiken.**
**Seite 4**




**Spielwitz, viele Level und tolle Grafik bietet »Crillion«. Gefragt sind schnelles Reaktions-vermögen und Ausdauer.**
**Seite 48**



Alle Programme aus Artikeln mit einem Disketten-Symbol finden Sie auf der beiliegenden Diskette (Seite 35).




**Grafiken auf den Bildschirm zaubern – das gehört zu den schönsten Dingen, die man mit einem C64 anstellen kann. Mit der schnellen Basic-Erweiterung »Grafik 2001« lassen sich diese programmieren.**

Der Basic-Interpreter des C64 kann eine ganze Menge, eines jedoch nicht: bequem und vor allen Dingen schnell Kreise, Linien und Rechtecke (Boxen) auf den Hires-Bildschirm zeichnen. Wer nichts mit Maschinensprache »am Hut« hat, muß auf unübersichtliche PEEK- und POKE-Befehle ausweichen.

Dies ändert sich schlagartig mit der Basic-Erweiterung »Grafik 2001«. Laden Sie das Programm mit

```
LOAD "GRAFIK 2001",8
```

und starten es mit RUN.

Nach der Initialisierung stehen Ihnen 41 neue Basic-Befehle zur Verfügung (Tabelle 1). Da sie in Form von Tokens gespeichert sind, lassen sie sich beim Programmieren in gewohnter Weise abkürzen. Nach THEN ist im Gegensatz zu anderen Basic-Erweiterungen kein Doppelpunkt nötig. Ein Beispiel:

```
IF A = 7 THEN GOSUB 1000
```

Die übrigen, bereits bekannten Befehle des Basic 2.0 behalten uneingeschränkt Gültigkeit.

Jede Bequemlichkeit hat ihren Preis: Durch die Einbindung der neuen Befehle in den Interpreter und dem Einrichten von drei voneinander unabhängigen Grafikbildschirmen wird der frei verfügbare Basic-RAM-Speicher auf 26623 Byte reduziert. Die letzte verfügbare Speicherstelle des Basic-Speichers wurde durch Manipulation der Adressen 55 und 56 auf $6FFF (28671) herabgesetzt. Dadurch können mit »Grafik 2001« Programme bis maximal 104 Blocks entwickelt und auf Diskette gespeichert werden.

Die Grafikerweiterung unterstützt die farbige Hiresgrafik des C64. Diese besteht aus 320 x 200 einzeln ansprechbaren Bildpunkten (Pixel). Die Lage dieser Pixel wird mit einer Koordinatenangabe für die horizontale (X) und vertikale (Y) Position bestimmt. In der linken, oberen Ecke des Grafikbildschirms befinden sich die Koordinaten 0/0, in der rechten unteren die Werte 319/199.

Die Farbinformation für je 8 x 8 Bildpunkte liegt im Video-RAM. Die oberen 4 Bit (High-Nibble) speichern die 16 möglichen Farbwerte für die Vordergrundfarbe (gesetztes Pixel), die unteren 4 Bit (Low-Nibble) sind für die Hintergrundfarbe (gelöschter Bildpunkt) zuständig. Damit steht fest, daß in einem 8 x 8-Punkte-Raster lediglich zwei verschiedene Farben vorkommen können. Allerdings bietet das Programm eine Funktion, mit der gezielt Bytes des Video-RAM farblich verändert werden. Auf diese Weise ist ein nachträgliches Einfärben einer Hires-Grafik möglich. Davon später mehr.

Was leisten die zusätzlichen Basic-Befehle? Wir werden sie im folgenden Abschnitt erläutern. Notwendige Parameterangaben sind zwingend und mit Variablen bezeichnet. Die Beschreibung dazu folgt im jeweiligen Erläuterungstext. Erlaubte Abkürzungen der Befehle stehen in Klammern. Der Großbuchstabe wird mit der SHIFT-Taste eingegeben.

**CLEAR (clE)**
Löscht den gesamten Grafikbildschirm. Vorder- und Hintergrundfarben werden nicht verändert.

**MODE n (mO)**
n = 1: schaltet den hochauflösenden Grafikmodus ein.
n = 0: aktiviert den Textbildschirm bei gleichzeitigem Einschalten des Großschrift/Blockgrafikmodus.

**COLOR p,h (coL)**
Setzt die Farben im Video-RAM (p = Pixelfarbe, h = Hintergrund). Es können Werte zwischen 0 und 15 angegeben werden.

**CHANGE (be) (chA)**
Vollzieht den Wechsel des Speicherinhalts zwischen drei Grafikbildschirmen. Die jeweiligen Bildpunkte werden ausgetauscht.
be = 0: (bzw. ohne Angabe): Inhalt Grafikschirm 1 nach 2.
be = 1: Speicherinhalt von Grafikschirm 1 nach 3.

Die Lage der einzelnen Hires-Bildschirme im Speicher:

**Grafikschirm 1: ab $E000 (57344)**
**Grafikschirm 2: ab $A000 (40960)**
**Grafikschirm 3: ab $7000 (28762)**

**INVERS (inV)**
Der sichtbare Grafikbildschirm wird revers dargestellt.

**COMB n (coM)**
Diese Anweisung führt eine logische Verknüpfung der Grafikbildschirmseiten aus (nach den Richtlinien der Boole'schen Algebra). Die Verknüpfungsart wird durch den Parameter »n« bestimmt:
n = 1: OR　　n = 2: AND　　n = 3: EXOR

**GSAVE a$, dv (gS)**
Speichert einen **verdeckten** Grafikbildschirm mit anzugebendem Filenamen, in Anführungsstrichen oder als Variable definiert (z.B. A$), auf die Floppy-Station mit der Gerätenummer »dv«.

**CSAVE a$, dv (cS)**
Die Farben des Video-RAM zur Grafik werden gespeichert. Die Parameter sind analog zu GSAVE einzugeben.

**GLOAD a$,dv (gL)**
Damit kann ein mit GSAVE gespeichertes Grafikbild oder ein mit CSAVE auf Diskette abgelegtes Video-RAM wieder

**»Grafik 2001« (Befehlsübersicht)**

| | | | |
|---|---|---|---|
| BEEP | DISCREEN | INVERSE | RORECHT |
| BLOCK | DUPLICATE | LINE | SCROLL |
| CHANGE | ELLIPSE | LOWCOL | SLOAD |
| CIRCLE | ENLARG | MARK | SPRITE |
| CLEAR | FILL | MIRROR | SPRPLOT |
| COLOR | GLOAD | MODE | SSAVE |
| COLPLOT | GSAVE | PLOT | TEST |
| COMB | GTSCREEN | POINT | TSCREEN |
| CSAVE | HCOPY | PSCLINE | WINDOW |
| CTEST | HVLINE | PSCREEN | |
| DIR | | RECHT | |

**Tabelle 1. »Grafik 2001« erweitert das Basic 2.0 des C64 um 41 neue Befehle**

geladen werden. Beide Ladeanweisungen sind nacheinander auszuführen.

**POINT zm,x,y (poI)**
Setzt, löscht oder invertiert einen Punkt an der mit x/y bezeichneten Koordinate. Der Parameter »zm« kann folgende Bedeutungen haben:
zm = 0: Punkt setzen
zm = 1: Punkt löschen
zm = 2: Punkt invertieren

**Achtung: Diese Angaben zu »zm« gelten für alle Anweisungen der Befehlsbeschreibung zu »Grafik 2001«.**

**PSCLINE x,y (psC)**
Löscht ab den angegebenen Koordinaten eine senkrechte Linie bis zum unteren Rand der Grafik.

**MARK mf,x,y (mA)**
Zeichnet eine Linie von fünf Pixeln Länge. Der Mittelpunkt dieses Striches wird durch die Koordinaten x/y bestimmt. Die Parameterangabe »mf« gibt die Richtung an:
mf = 0: horizontal
mf = 1: vertikal

**HVLINE hv,z (hV)**
Zieht eine Linie über den gesamten Grafikbildschirm, je nach Inhalt des Parameters »hv«:
hv = 0: horizontal hv = 1: vertikal

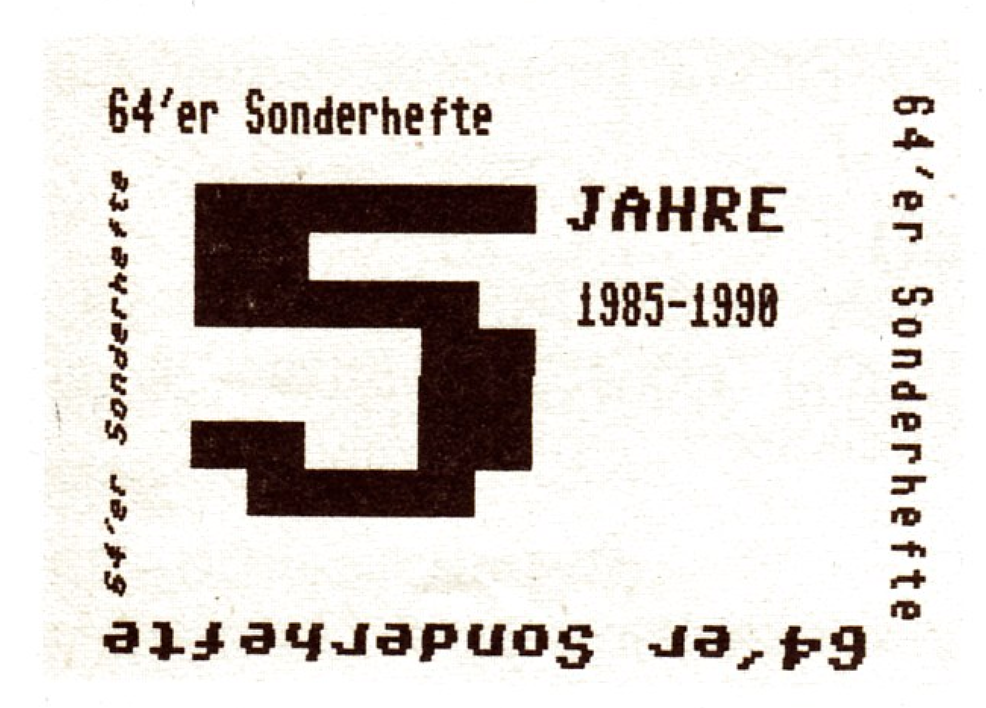



**Bild 1. Die TEXT-Anweisung bietet unzählige Variationen zum Beschriften einer Grafik**

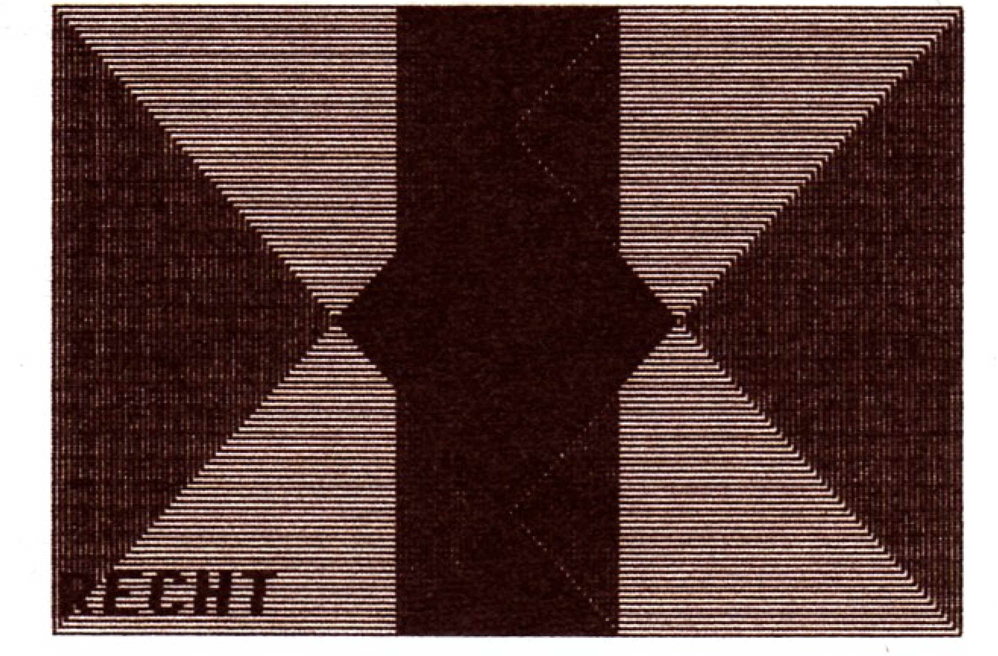



**Bild 2. Beispiel einer mit dem Befehl RECHT gezeichneten Grafikfigur**

Der weitere Parameter »z« definiert entsprechend dem eingestellten Wert für »hv« die X- oder Y-Position, durch die der Strich gezogen wird.

**LINE zm,x1,y1,x2,y2 (liN)**
Eine durchgehende Linie wird von einer beliebig wählbaren Anfangskoordinate (x1/y1) zur Endkoordinate (x2/y2) gezogen. Dabei spielt es keine Rolle, ob der Strich waagrecht, senkrecht oder diagonal über den Grafikbildschirm verläuft.

**CIRCLE zm,x,y,rx,ry (cI)**
Ein Kreis bzw. eine Ellipse wird um den Kreismittelpunkt x/y mit horizontalem Radius rx und vertikalem Radius ry gezeichnet. Sind beide Radienwerte gleich, erhalten Sie einen Kreis und keine Ellipse.

**ELLIPSE zm,x,y,rx,ry,dw,sw,ew,s (eL)**
Hier handelt es sich um eine Modifizierung des Befehls CIRCLE. Ein Ellipsenbogen mit dem Mittelpunkt x/y und den Radien rx/ry wird gezogen.

**dw :** Kennzeichnet den Drehwinkel der Ellipsenhauptachse um den Mittelpunkt. Der Ellipsenbogen wird im Uhrzeigersinn um den Mittelpunkt gedreht.
**sw, ew :** Start- und Endwinkel des Ellipsenbogens
**s :** Gibt den Schrittwinkel an, mit dem die einzelnen Bogenpunkte errechnet werden. Je kleiner dieser ist (Werte unter »1«, immer größer »0« sind möglich), desto kompakter wird die Ellipsengrafik gezeichnet.

Einzelne Punkte im Ellipsenbogen werden errechnet und mit einer Linie verbunden. Die Winkel müssen im Bogenmaß angegeben werden. Bogenmaß bedeutet die Größe eines Kreisbogens, gemessen mit dem Mittelpunktswinkel »arc« (Abk. griechischer Kleinbuchstabe »Alpha«) und dem Halbmesser 1, der sich aus folgender Formel errechnet:

```
Ludolf'sche Zahl pi * arc / 180
```

Drehungen führt die Routine im Uhrzeigersinn aus.

**TEXT zm,r,b,h,v,a,x,y,a$ (tE)**
Mit diesem Befehl beweist »Grafik 2001« seine Vielseitigkeit (Bild 1). Ein beliebiger Text (ob direkt oder als String definiert) wird in den Grafikbildschirm geschrieben. Die Stärke des Befehls sind die vielen Möglichkeiten der Textgestaltung. Dazu werden einige Parameterangaben notwendig:
**r = 1:** von links nach rechts (Normalfunktion)
**r = 2:** von rechts nach links
**r = 3:** von unten nach oben
**r = 4:** von oben nach unten

Die Schrift wird, der Richtung entsprechend, gekippt ausgegeben. Dies bedeutet, daß z.B. beim Parameterwert r = 2 der Text auf dem Kopf stehend erscheint.
**b :** Breite des einzelnen Zeichens. Werte von 1 (normal) bis 25 sind möglich.
**h :** Zeichenhöhe (von 1 bis 25)
**v :** Darstellung in Kursivschrift. Dieser Parameter bestimmt die Verschiebung eines Zeichens von der Spitze zum Fuß. Werte von 0 (normal) bis 8 (45 Grad Schräge) sind einstellbar.
**a :** Abstand der Zeichen. Bei normaler Zeichenbreite (b = 1) ist der einzutragende Wert »8«.
**x und y** bezeichnen die Startkoordinaten (linke obere Ecke des ersten Zeichens eines Strings).
**A$ :** String, der ausgegeben werden soll. Die Zeichenkette muß nicht unbedingt in einer Variablen festgelegt werden. Ein Text-String am Ende der Anweisung muß in Hochkommas stehen. Es ist nicht notwendig, numerische Werte in einen String umzuwandeln (STR$-Funktion). Sie werden von dieser Routine ebenfalls akzeptiert. Innerhalb der Zeichenkette können Sie mit Hilfe der <CTRL>-Taste die Darstellung des Zeichensatzes wechseln:

**<CTRL> <9>** Revers-Modus ein
**<CTRL> <0>** Revers-Modus aus
**<CTRL> <A>** Großschrift/Blockgrafik
**<CTRL> <B>** Klein-/Großschrift

Der Großschrift/Blockgrafik-Modus ist nach dem Programmstart voreingestellt.

**FILL x,y (fI)**
Malt eine durch Linien oder den Bildschirmrand begrenzte Figur in der mit COLOR gewählten Vordergrundfarbe aus. X/Y sind die Startkoordinaten der Füllroutine, wobei Sie unbedingt darauf achten müssen, daß diese sich innerhalb eines begrenzten Bereichs befinden. Andernfalls wird der gesamte Bildschirm übermalt.

**DUPLICATE x1,y1,x2,y2,x3,y3 (dU)**
Ein durch die Koordinaten x1/y1 (links oben) und x2/y2 (rechts unten) ausgewählter Grafikausschnitt wird originalge-

treu an die mit x3/y3 (linke obere Ecke) festgelegte Position innerhalb des Bildschirms kopiert.

**SCROLL r,x1,y1,x2,y2 (scR)**
Rolliert einen mit den oberen (x1/y1) und unteren Koordinatenpunkten (x2/y2) ausgewählten Teil der Grafik um einen Bildpunkt in die durch den Parameter »r« festgelegte Richtung. Die einzelnen Richtungswerte bedeuten:
r = 1: Scrollen nach rechts
r = 2: Scrollen nach links
r = 3: Scrollen nach oben
r = 4: Scrollen nach unten
Effektvoll nutzen können Sie diesen Befehl innerhalb einer FOR-NEXT-Schleife.

**WINDOW y1,y2 (wI)**
Diese Anweisung erzeugt einen sog. »Split-Screen«, die gleichzeitige Bildschirmdarstellung von Grafik- und Textmodus. Die Begrenzungsrasterzeile der beiden Modi ist innerhalb des Koordinatensystems 319 x 199 frei wählbar. Der Beginn des Grafikbildschirms wird mit der vertikalen Zeile y1, der untere Rand mit y2 bestimmt. Der darunterliegende Bereich ist Textbildschirm.

**LOWCOL x,y,p,h (loW)**
Damit können Sie gezielt ein Byte des Video-RAM ändern. Dies wirkt sich auf die Farbgebung eines 8 x 8-Pixel-Rasters innerhalb des Grafikbildschirms aus. Der linke, obere Beginn dieses Bildpunkte-Bereichs wird durch die Koordinaten x/y festgelegt. Die Parameter »p« und »h« bestimmen die Vorder- und Hintergrundfarbe (siehe Beschreibung zu COLOR).

**COLPLOT p,h (colP)**
Diese Anweisung ermöglicht ein mehrfarbiges Zeichnen in der Hires-Grafik, wobei die Parameter »p« (Vordergrundfarbe) und »h« (Hintergrund) jeweils Werte zwischen »0« und »15« annehmen können. Die Eingabe von »COLPLOT« ohne Parameter schaltet diese Option ab.

**PLOT n (pL)**
Arbeitet unabhängig von COLPLOT nach dem Prinzip der Anweisung PSCLINE.
n = 2: Unter jedem gezeichneten Punkt wird eine Linie bis zum Bildschirmrand gelöscht bzw. in der Hintergrundfarbe gezeichnet. Damit lassen sich effektvolle 3D-Bilder erzielen.
n = 1: Die normale Punktsetz-Funktion wird wieder aktiviert.
Hinweis: Invertierte Grafikbefehle (zm = 2) werden weder von COLPLOT noch von PLOT beeinflußt.

**RECHT zm,x1,y1,x2,y2 (reC)**
Zeichnet ein Rechteck mit den linken, oberen Koordinaten x1/y1 und den rechten, unteren Endpunkten x2/y2 auf den Grafikbildschirm. Ist x1 identisch mit y1 und x2 mit y2, entsteht ein Quadrat. Bei geschickter Programmierung lassen sich interessante Bilder erzeugen (Bild 2).

**BLOCK zm,x1,y1,x2,y2,ri,sa (bL)**
Prinzipiell arbeitet dieser Befehl wie RECHT. Das Rechteck kann im Gegensatz zu FILL schraffiert dargestellt werden. Zwei neue, optionale Parameterangaben sind dazugekommen:
ri = 0: horizontale Schraffierung
ri = 1: vertikale Schraffierung
»sa« legt den Abstand der Schraffur fest. Bei sa = 1 wirkt BLOCK wie die FILL-Anweisung.

**RORECHT zm,x,y,wi,a1,a2 (rO)**
Dieser Grafikbefehl bringt ebenfalls ein Rechteck auf den Bildschirm. X/Y bedeuten den linken, oberen Eckpunkt im Bogenmaß. Der entsprechende Neigungswinkel wird mit »wi« festgelegt, »a1« und »a2« bezeichnen die horizontale und vertikale Seitenlänge. Rechtecke oder Quadrate lassen sich mit dieser Anweisung je nach Angabe des Parameters »wi« schräg gekippt darstellen.

**PSCREEN n (pscR)**
n = 2: Alle Befehle, die eine Hires-Grafik verändern, wirken sich nicht auf die sichtbare, sondern auf die zweite, unsichtbare Grafik aus.
n = 1: Der Normalzustand ist hergestellt. Die Zeichenbefehle beeinflußen wieder den sichtbaren Grafikbildschirm.

**TSCREEN n (tS)**
n = 2: Beeinflußt die Anweisungen SPRITE, SCROLL, DUPLICATE, TEST und CTEST. Sie wirken sich in einer unsichtbaren Grafik aus. Man kann z.B. mit DUPLICATE Ausschnitte in den anderen Grafikbildschirm übertragen.
n = 1: schaltet um zur sichtbaren Grafik.

**DISCREEN x1,y1,p (diS)**
Ein Ausschnitt von 40 x 20 Bildpunkten aus dem Hires-Bildschirm wird in vergrößerter Form auf den »normalen« Textbildschirm (MODE 0) übertragen. Zur Anzeige eines gesetzten Punktes verwendet das Programm das Blockgrafikzeichen CHR$(209). Die Ausschnittvergrößerung benötigt 40 Spalten und 20 Zeilen auf dem Textbildschirm. Der Parameter »p« ist optional und kann ganz entfallen, wenn sein Wert »0« beträgt. Bei p=1 werden 40 x 78 Bildpunkte übertragen, was aber weniger für den Bildschirm, sondern für eine Druckerausgabe sinnvoll ist.

**GTSCREEN zm,x1,y1 (gT)**
Dieser Befehl wirkt umgekehrt zu DISCREEN: Eine Blockgrafik (bestehend aus CHR$(209)-Zeichen) wird an den Hires-Koordinaten x1/y1 in den Grafikbildschirm eingefügt.

**ENLARG em (enL)**
Aktiviert eine Vergrößerung des linken, oberen Quadranten eines Hires-Bildes. Die Bedeutung des Parameters »em«:
em = 0: Vergrößerung in X- und Y-Richtung
em = 1: nur in X-Richtung (horizontal)
em = 2: nur in Y-Richtung (vertikal)

**MIRROR hv,sz,of (miR)**
Der Grafikbildschirm wird gespiegelt. Durch geschickte Anwendung der Parameterwerte lassen sich eindrucksvolle Effekte erzielen:
hv = 0: Spiegelung an der Y-Achse.
hv = 1: Spiegelung an X- und Y-Achse.
»sz« enthält die Anzahl der zu spiegelnden Zeilen bzw. Spalten (Berechung nach den Textbildschirm-Koordinaten 40 x 25).
»of« ist für das Aussehen der Grafik nach dem Spiegelvorgang zuständig:
of = 0: vollständige Spiegelung
of = 1: OR-Verknüpfung mit dem ursprünglichen Bild.

## Kurzinfo: Grafik 2001

**Programmart:** Erweiterung des Basic 2.0
**Laden:** LOAD "GRAFIK 2001",8
**Starten:** Nach dem Laden RUN eingeben
**Besonderheiten:** Die neuen Basic-Anweisungen können im Direktmodus und in einem Programm benutzt werden. Mit »Grafik 2001« erzeugte Basic-Programme sind nur innerhalb dieser Basic-Erweiterung lauffähig und dürfen nicht länger als maximal 104 Blocks auf Diskette sein. Die Hardcopy-Routine (HCOPY) ist auf die Commodore-Drucker MPS 801/803 und Kompatible abgestimmt.
**Benötigte Blocks:** 42 Blocks
**Programmautoren:** Frank-Rüdiger Brendel, Dr. Heinz Domes

**SPRITE n,x,y (sprl)**
Mit dieser Anweisung wird ein Sprite (24 x 21 Pixel) aus einem Teilbereich des aktuellen Grafikbildschirms gebildet. »n« kann Werte von 0 bis 7 annehmen und definiert die Nummer des Datenblocks (120 bis 127), in dem das jeweilige Spritemuster abgelegt ist. Die Zahlen dieser Blöcke müssen in den entsprechenden Sprite-Zeigeradressen 50168 bis 50175 mit POKE gesetzt werden. Das Herausfiltrieren des Sprites aus der Grafik funktioniert im Prinzip wie bei DUPLICATE: X/Y geben die linke, obere Ecke des Grafikausschnittes an. Dazu noch ein Hinweis: Auf diese Weise erzeugte Spriteformen gelten nur im Grafikmodus (MODE 1). Im Textmodus (MODE 0) müssen Sie die üblichen Basic-POKE-Anweisungen zum Definieren und Erstellen von Sprites verwenden.

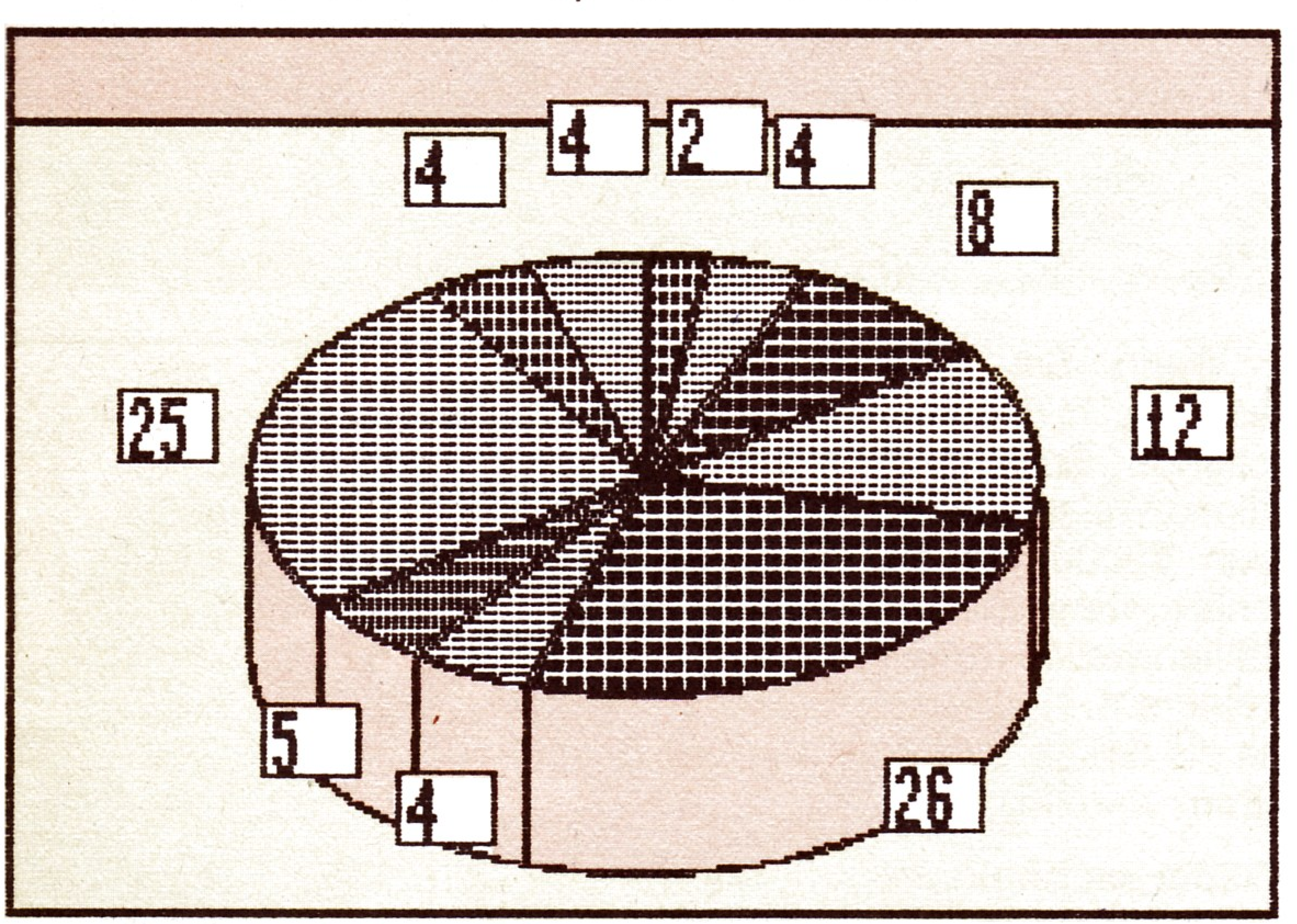


**Bild 3. Statistische Präsentationsgrafiken lassen sich eindrucksvoll darstellen**

**SSAVE n,a$,dv (sS)**
Speichert einen Spriteblock mit der Nummer »n« und dem Filenamen »A$« auf Diskette in der Floppy-Station »dv« ab.

**SLOAD n,a$,dv (slO)**
Lädt einen mit SSAVE gespeicherten Spritemuster-Block der Nummer »n« mit dem angegebenen Filenamen »A$« von der Floppy »dv«.

**SPRPLOT n,zm,x,y (spR)**
Fügt ein Sprite mit der Nummer »n« an den linken, oberen Koordinaten x/y in ein Hires-Bild ein.

**TEST (x,y) (tE)**
Dient zur Überprüfung eines gesetzen Pixels an der Koordinatenposition x/y. Das Ergebnis der Funktion TEST sollte in einer Variablen gespeichert werden, z.B. TE. Ein Beispiel:
Sie wollen testen, ob an Position X=160 und Y=100 ein Bildpunkt gesetzt ist.

```
TE=TEST(160,100):PRINT TE
```

Hat die Variable TE den Wert »1«, ist an dieser Stelle ein Punkt gesetzt. Andernfalls erscheint als Ergebnis »0«.

**CTEST (x,y) (cteS)**
Weist einer Variablen, z.B. CT, den Wert (0 - 15) der sichtbaren Farbe eines Bildpunktes zu. Beispiel:

```
CT = CTEST(200,180):PRINT CT
```

Die Vordergrundfarbe an der Position 200/180 wird in der Variablen CT gespeichert.

**DIR**
Sorgt für die Auflistung des DIRECTORY einer Diskette auf dem Bildschirm, ohne ein im Speicher stehendes Basic-Programm zu löschen. Das Auflisten des Inhaltsverzeichnisses kann mit der Taste <CTRL> verlangsamt, jedoch nicht mit <STOP> abgebrochen werden.

**BEEP tf (bE)**
Erzeugt einen Signalton mit variabler Tondauer und Frequenz. Der Parameter »tf« ist optional und kann Werte von »0« bis »255« annehmen.

**HCOPY vg (hC)**
Bewirkt einen Grafikausdruck mit einem MPS 801, MPS 803 und ähnlichen Druckern, die gleichartige Steuercodes zum Drucken einer Grafik verstehen. Dies erledigen Hardware-Interfaces wie z.B. Wiesemann, Görlitz, Merlin usw. Der Zusatz »vg« hat beim Ausdruck folgende Bedeutung:
vg = 0: 200 x 320 Punkte
vg = 1: 200 x 640 Punkte (horizontal vergrößert)
vg = 2: 400 x 320 Punkte (vertikal vergrößert)
vg = 3: 400 x 640 Punkte (in beiden Richtungen vergrößert)

Sollten Sie mit Ihrem Drucker keinen Erfolg haben, speichern Sie die Hiresgrafik ab und erledigen den Ausdruck mit einem separaten Grafik-Druckprogramm, das speziell auf die Sequenzen Ihres Druckers zugeschnitten ist.

**HINWEISE ZUM PROGRAMM**
Hat sich bei einem mit »Grafik 2001« erstellten Basic-Programm ein Fehler eingeschlichen, schaltet es beim Programmablauf von MODE 1 in MODE 0 und gibt die entsprechende Fehlermeldung aus. Der WINDOW-Befehl wird inaktiviert. Dies geschieht ebenfalls beim Betätigen der Tastenkombination <RUN/STOP RESTORE>, da die Basic-Erweiterung über eine eigene NMI/BREAK-Routine verfügt. Die vom Programm vorgesehenen Farben für Rahmen, Hintergrund (beide blau) und Schrift (hellblau) werden erneut gesetzt. Wenn Ihnen die Farbkombinationen nicht gefällt, können Sie diese ändern: Geben Sie nach dem Laden von »Grafik 2001«, aber **vor** dem Starten mit RUN, folgende POKE-Anweisungen im Direktmodus ein:

```
POKE 3111, Farbzahl (0 - 15) für Hintergrund
POKE 3119, Farbzahl (0 - 15) für Schrift
```

Falls das Programm nach der Änderung auf Diskette gespeichert wird, steht Ihnen bei künftiger Verwendung der Basic-Erweiterung diese Farbgebung zur Verfügung. Möchten Sie die Farben nach aktivierter Basic-Erweiterung abändern, funktioniert dies mit folgenden Anweisungen im Direktmodus:

```
POKE 51079, Farbzahl (0 - 15) für Hintergrund
POKE 51087, Farbzahl (0 - 15) für Schriftzeichen
```

Wenn Sie die RESTORE-Taste drücken, werden die gewünschten Farben aktiviert.

**Demo-Programme auf Diskette**
Um Ihnen die Wirkungsweise verschiedener Grafikbefehle vor Augen zu führen, finden Sie eine Anzahl Demos auf der beiliegenden Diskette. Sie müssen mit der üblichen Ladeanweisung geladen und mit RUN gestartet werden. Beim File »GRA1« handelt es sich um ein Ladeprogramm. Es lädt innerhalb eines Menüs zwei weitere Demoprogramme:
- TESTPROGRAMM, das in anschaulicher Weise die grafischen Möglichkeiten von »Grafik 2001« zeigt, und
- STATISTIK, ein Beispiel für eine Anwendung mit den neuen Grafikbefehlen (Bild 3).

Die übrigen Beispielprogramme (die jeweiligen Filenamen beginnen mit dem Wort »DEMO.«) geben Ihnen einen Überblick, wie die neuen Basic-Anweisungen von »Grafik 2001« effektvoll genutzt und eingesetzt werden können.
Wenn Ihnen die Arbeitsweise mit dieser professionellen Basic-Erweiterung in »Fleisch und Blut« übergegangen ist, werden Sie feststellen, daß »Grafik 2001« dem C64 Befehle zur Verfügung stellt, von denen andere vergleichbare Computer (auch ein C128) nur träumen können.

(Frank-Rüdiger Brendel/Dr. Heinz Domes/bl)

**Egal wo sie liegt – »Hardmaker« findet jede Hiresgrafik im Speicher des C64. Das Programm kann noch mehr: Es druckt sie auch aus.**

# AUF DER SUCHE NACH DER GRAFIK

Hochauflösende Grafikbilder aus nahezu allen Programmen können Sie mit »Hardmaker« zu Papier bringen. Umfangreiche Routinen stehen zur Verfügung. Multi-Color-Bilder können vor dem Ausdruck sogar in Graustufen umgerechnet werden.

### Grafiken aus Programmen ausfiltern

Laden Sie das gewünschte Programm, aus dem »Hardmaker« Grafikbilder aussondern soll (z.B. ein Spiel) und starten es. Erscheint das Hiresbild auf dem Bildschirm, brechen Sie das laufende Programm ab (entweder mit <RUN/STOP RESTORE> oder durch einen RESET). Der Computer muß sich wieder im Direktmodus befinden. Laden Sie das Utility von der beiliegenden Diskette mit

```
LOAD "HARDMAKER",8
```

und starten es mit RUN.

Auf dem Bildschirm erscheint der Speicherbereich von $2000 (8192) bis $3FFF (16383), als Multi-Color-Grafik dargestellt. Hatte das vorher geladene Programm in diesem Bereich ein Hiresbild generiert, sehen Sie es auf dem Bildschirm. Der genannte Bereich wird von »Hardmaker« als Grafik-RAM benutzt. Andernfalls finden Sie darauf lediglich ein wirres Durcheinander von Punkten.

### Speicherbereiche:

Computergrafiken können nur an bestimmten Stellen im Speicher stehen, um vom VIC ausgelesen werden zu können. Ein solcher Bereich ist der von $2000 bis $3FFF. Diesen Bereich sehen Sie grundsätzlich auf dem Bildschirm; er wird vom Programm als Grafik-RAM benutzt. Wollen Sie den Inhalt eines anderen Bereiches sehen, muß er nach $2000 transportiert werden. Dazu dienen die Tasten ⟨1⟩ bis ⟨6⟩ und ⟨←⟩:

| | |
|---|---|
| ⟨1⟩ : | $4000 bis $5FFF |
| ⟨2⟩ : | $6000 bis $7FFF |
| ⟨3⟩ : | $8000 bis $9FFF |
| ⟨4⟩ : | $A000 bis $BFFF (RAM unterm Basic) |
| ⟨5⟩ : | $C000 bis $DFFF ($D000 bis $DFFF; RAM unter I/O) |
| ⟨6⟩ : | $E000 bis $FFFF (RAM unterm Kernel) |
| ⟨←⟩: | $0000 bis $1FFF, Dieser Bereich ist nur der Vollständigkeit halber per Taste erreichbar. Benutzen können Sie ihn nicht, da dort Zeropage, Stack, Video-RAM und der Hardmaker selbst liegen! |

Wenn Sie auf eine dieser Tasten ohne ⟨SHIFT⟩, ⟨CTRL⟩ oder ⟨CBM⟩ drücken, wird der entsprechende Speicherbereich nach $2000 transportiert und ist damit auf dem Bildschirm sichtbar. Drücken Sie jedoch ⟨SHIFT⟩ und eine dieser Tasten, wird der entsprechende Bereich mit dem ab $2000 ODER-verknüpft. So können zwei Bilder zusammengemischt werden. Das Ergebnis liegt wieder ab $2000 im Speicher.

Folgende Kombinationen bewirken also:

| | |
|---|---|
| ⟨←⟩ bis ⟨6⟩ | ≙ jeweiligen Bereich nach $2000 kopieren |
| ⟨SHIFT⟩ | ≙ ODER-Verknüpfen |
| ⟨CBM⟩ | ≙ EX-OR-Verknüpfen |
| ⟨SHIFT CBM⟩ | ≙ UND-Verknüpfen |
| ⟨CTRL⟩ | ≙ Bereich mit dem ab $2000 vertauschen |
| ⟨CTRL SHIFT⟩ oder ⟨CTRL CBM⟩ | ≙ $2000 bis $3FFF in entsprechenden Bereich kopieren |

Wenn Sie eine Kombination mit <CTRL> drücken, wird der entsprechende Speicherplatz verändert. Auf diese Weise können Sie z.B. den Inhalt von $2000 bis $3FFF zwischenspeichern, wenn Sie ihn danach weiterbearbeiten wollen (etwa bei schwierigen Korrekturen). Nach erfolgter Änderung sollte die fertige Grafik auf Diskette gespeichert werden.

### Bilder »schneiden«:

Manchmal kommt es vor, daß nicht der gesamte Inhalt des Bildschirms zu einer Grafik gehört und man den Rest »wegschneiden« möchte. Dazu drücken Sie die Taste <R>. Der Rahmen wechselt seine Farbe und an der rechten Seite erscheint eine flackernde Linie, die Sie mit <CURSOR links/rechts> hin- und herbewegen können.

Wenn Sie die Linie maßgerecht und korrekt positioniert haben, drücken Sie





**Bild 1. Eine Hardcopy ohne Umrechnung...**

auf ⟨SPACE⟩; der Bereich rechts der Linie wird gelöscht (oder gefüllt, wenn Sie ⟨SHIFT SPACE⟩ drücken). Möchten Sie nichts löschen, dann drücken Sie ⟨Q⟩, und die flackernde Linie ist verschwunden.

Im Schneidemodus stehen folgende Funktionen zur Verfügung:
— Cursor nach oben/unten (nur ⟨O⟩, ⟨U⟩)
— Cursor nach oben links/rechts (nur ⟨L⟩, ⟨R⟩)
— ⟨SPACE⟩, ⟨SHIFT SPACE⟩
— ⟨Q⟩ (wie quit)

| | | |
|---|---|---|
| ⟨R⟩ : | rechts | Randfarbe: orange |
| ⟨L⟩ : | links | Randfarbe: blau |
| ⟨O⟩ : | oben | Randfarbe: hellrot |
| ⟨U⟩ : | unten | Randfarbe: grün |

**Bilder verschieben:**
Wenn die Grafik nicht genau oben links beginnt, muß sie verschoben werden. Eine Möglichkeit dazu ist das Scrollen:
Mit CBM-Taste + Cursor-Taste wird die Grafik um 1 Byte nach links oder rechts verschoben.



**Bild 2. ...und eine mit Umrechnung der Multi-Color-Daten**

Die andere Möglichkeit sind die Tasten ⟨A⟩ und ⟨E⟩. Positionieren Sie den Cursor irgendwo mitten auf dem Bildschirm und drücken Sie ⟨A⟩: Die Grafik wird so verschoben, daß die Cursorposition nun den Anfang der Grafik bildet. Analog funktioniert hier die Taste ⟨E⟩: Die Cursorposition bildet jetzt das Ende der Grafik. Diese Funktionen ermöglichen ein bequemes Positionieren einer Grafik, die irgendwo im Speicher liegt.

**Farbe:**
Mit den Funktionstasten kann die Farbe der Grafik geändert werden, wenn die Ausgangsbelegung (die sich für Schwarzweiß-Fernseher übrigens gut eignet) nicht gefällt:

| | |
|---|---|
| ⟨F1⟩/⟨F2⟩ ≙ | Farbe 1 HiRes/Multi |
| ⟨F3⟩/⟨F4⟩ ≙ | Farbe 2 HiRes/Multi |
| ⟨F5⟩/⟨F6⟩ ≙ | Farbe 3 Multi |
| ⟨F7⟩/⟨F8⟩ ≙ | Farbe 4 Multi |

Die Funktionstaste blättert die Farben vorwärts, geSHIFTet blättert sie die Farben zurück. Für Multi und HiRes sind getrennte Farb-Speicher vorhanden.

**Folgende Tasten bewegen den Cursor:**

| | |
|---|---|
| Cursortasten | gewohnte Cursor-Bewegung |
| ⟨RETURN⟩ | Bewegt ihn in die erste Spalte der nächsten Zeile |
| ⟨SHIFT CLEAR/HOME⟩ | löscht die Grafik |
| ⟨CLEAR/HOME⟩ | Cursor links oben |
| ⟨£⟩ | (Gegenteil von HOME: Cursor in letzte Spalte letzte Zeile) |
| ⟨SPACE⟩ | Cursor rechts + Cursorfeld löschen |
| ⟨SHIFT SPACE⟩ | Cursor rechts + Cursorfeld füllen mit Farbe 3 |
| ⟨CBM SPACE⟩ | Cursor rechts + Cursorfeld füllen mit Farbe 1 |
| ⟨CTRL SPACE⟩ | Cursor rechts + Cursorfeld füllen mit Farbe 2 |
| ⟨INST/DEL⟩ | genauso wie SPACE, bloß mit Cursor nach links (also kein echtes DEL!) |

**Sonderfunktionen:**

| | |
|---|---|
| ⟨H⟩ | schaltet HiRes-Modus ein; genauso wie Sie die Grafik nun sehen, wird sie von einem Matrixdrucker ausgegeben |
| ⟨M⟩ | schaltet Multicolor an. |
| ⟨T⟩ | ist eine sehr praktische Sache, wenn man einen der unter »Hardcopy« erwähnten Matixdrucker besitzt. Diese geben die Multicolor-Grafiken nämlich so aus, daß die Farben 1 und 2 als charakteristische Linien erscheinen. T verwandelt diese Farben nun in Graustufen, die der Drucker ausgeben kann (siehe auch Textkasten). T funktioniert nur, wenn Multicolor eingeschaltet ist, und schaltet dann auf HiRes um! |
| ⟨I⟩ | invertiert die Grafik |
| ⟨S⟩ | spiegelt die Grafik an der Vertikalen, und vertauscht die Farben 1 und 2, die ja ebenfalls gespiegelt werden. Spiegeln an der Horizontalen ist nicht nötig, da man ja die Hardcopy einfach umdrehen kann! |
| ⟨X⟩ | kehrt ins Basic zurück. |

**Funktionsweise des T-Befehls beim Hardmaker**
Im Multicolor-Modus stellen immer je 2 Bit eines Bytes die Information für einen Bildpunkt zur Verfügung, so daß die Auflösung gegenüber dem HiRes-Modus halbiert wird.

Dafür hat man vier Farben statt zwei zur Auswahl, die durch die Bitkombinationen 00, 01, 10 und 11 repräsentiert werden. Ein normaler Matrixdrucker hat hingegen keinen Multicolor-Modus und stellt die Bitmuster genau so dar, wie sie im Grafik-ROM stehen. Das führt bei Flächen, die aus den Farben 1 und 2 beziehungsweise den Bitmustern 01 oder 10 bestehen, zu den bekannten Streifenmustern (Bild 1). Um diese in Graustufen (Bild 2) umzuwandeln, geht der Computer folgendermaßen vor: Die vier Zwei-Bit-Kombinationen eines jeden der 8000 Byte des Grafik-RAMs werden nach den Bitkombinationen 01 und 10 untersucht.
1. Fall: Zwei-Bit-Kombination = 00 oder 11: nichts wird verändert.
2. Fall: Zwei-Bit-Kombination = 01: In diesem Fall werden die zwei Bit invertiert, also durch 10 ersetzt, aber nur dann, wenn die zwei Bit Bestandteil des 1., 3., 5., etc. Bytes des Grafik-RAMs sind. Dadurch wird das Bitmuster in jeder zweiten Grafikzeile um ein Bit nach links versetzt. Wenn die ersten 3 Byte

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **01 01 00 01** | | | | | **10** | **10** | **00** | **10** |
| **01 01 11 01** | lauten, so wird daraus | | | | **01** | **01** | **11** | **01** |
| **01 01 01 01** | | | | | **10** | **10** | **10** | **10** |

und der Drucker gibt ein gleichmäßig graues Feld aus.
3. Fall: Bit-Kombination = 10

Hierbei würde aus den ersten 3 Byte

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 10 | 10 | 00 | 10 | | 01 | 10 | 00 | 10 |
| 10 | 10 | 11 | 10 | folgendes werden: | 10 | 01 | 11 | 01 |
| 10 | 10 | 10 | 10 | | 01 | 10 | 01 | 10 |

Also ein etwas gröberes Graufeld. Dieser Effekt wird dadurch erreicht, daß eine Zwei-Bit-Kombination nur dann invertiert wird, wenn sie entweder im 1., 3., 5., ... Byte die 1. oder 3. Zwei-Bit-Kombination oder im 2., 4., 6., ... Byte die 2. oder 4. Zwei-Bit-Kombination ist.

### Diskwork-Modus

Wird mit ⟨D⟩ aktiviert. Die Grafik wird dann ausgeblendet und Sie haben den normalen Kernel-Editor vor sich. Folgende Befehle wurden hier implementiert.

| | |
|---|---|
| $ | Directory |
| @ | Kommandokanal des Laufwerks abfragen |
| @i | initialisieren (analoges gilt für SCRATCH, RENAME, FORMAT, etc.) |
| n | gibt die aktuelle Geräteadresse aus (Voreingestellt 8) |
| n9 | schaltet auf Geräteadresse 9 um |
| ? | arbeitet wie der normale Basic-PRINT-Befehl, kann aber hier für Berechnung genutzt werden |
| Q | schaltet den Grafikmodus wieder an |
| X | Rückkehr zum Basic |
| ← | Mit ← kann aus dem Hardmaker heraus eine Grafik gespeichert werden; zum Beispiel »←Bild 1 ⟨RETURN⟩« speichert den Bereich $2000 bis $3FFF (genauer bis $3F40) auf Diskette unter dem Namen »Bild 1« ab. Das erzeugte Programmfile ist 32 Blöcke lang. |
| £ | Um ein Programm nach Grafiken zu durchsuchen, muß man es irgendwie in den Computer bekommen. Die eine Möglichkeit ist, das Programm zu laden, zu starten und nach Erscheinen der Grafik durch Betätigen des RESET-Tasters auszusteigen. Diese Methode haben wir bereits kennengelernt. Eine andere Vorgehensweise ist, erst den »Hardmaker« zu laden und mit <D> den Diskwork-Modus einzuschalten. Die Funktion <£> gilt als Ladeanweisung für ein beliebiges Grafikbild. Der Filename darf nicht in Anführungszeichen stehen, die Laufwerksbezeichnung muß weggelassen werden: £Name Die Grafik wird an den Bereich ab $2000 (8192) geladen. |

### Hardcopy

Dieser Programmteil ist zugleich der wichtigste wie auch der problematischste. Denn eine Hardcopy-Routine muß an die meisten Drucker speziell angepaßt werden. Die im »Hardmaker« integrierte Hardcopy-Routine spricht Drucker mit sieben Nadeln an, also den MPS 801, den MPS 803, und Epson-Drucker mit entsprechendem Interface. Für Letztere hilft nur ausprobieren oder eine andere Hardcopy-Routine. Diese kann ab der Adresse $13A0 an den »Hardmaker« angehängt werden. Dazu kommen wir aber später noch.

## Kurzinfo: Hardmaker

**Programmart:** Aussondieren von Grafikbildern im Computerspeicher. Ausgabe auf Drucker ist möglich.
**Laden:** LOAD "HARDMAKER",8
**Starten:** Nach dem Laden RUN eingeben
**Steuerung:** Tastatur
**Besonderheiten:** Druckertreiber-Programm »Multiprint« kann nachgeladen und angepaßt werden
**Benötigte Blocks:** 18 Blocks
**Programmautor:** Chr. Kurts

Aktiviert wird die Hardcopy mit der Taste P. Daraufhin färbt sich der Rahmen schwarz und der C 64 beginnt mit der Hardcopy. Danach wird in den Grafikmodus zurückgesprungen.

## »Hardmaker« für Epson-Drucker

Warum soll man sich mit einem Epson-Drucker auf die Grafikfähigkeiten eines MPS 801/803 beschränken? Die grafischen Möglichkeiten eines Epson-kompatiblen Druckers erlauben es außerdem, Multicolor-Grafiken durch Zuordnung von Graustufen auszugeben. Dadurch kann eine noch realistischere Wiedergabe in Graustufen (»T-Befehl«) geboten werden.

Für die Ausgabe von HiRes-Grafiken wurde bei »Multiprint« darauf Wert gelegt, daß sowohl Bilder im HiRes-Modus als auch Grafiken im Multicolor-Modus gedruckt werden können. Um eine möglichst große Flexibilität zu erreichen, wurden Hardcopy-Routinen gewählt, die sowohl Drucker über den seriellen IEC-Bus als auch über eine Centronics-Schnittstelle bedienen können.

An der Bedienung hat sich nichts geändert. Der Start des Druckerprogramms erfolgt durch Drücken des Taste ⟨P⟩. Dabei erfolgt der Ausdruck im HiRes-Modus, wenn die Bildschirm-Darstellung ebenfalls HiRes darstellt. Wurde zuvor mit Taste ⟨M⟩ Multicolor angewählt, so wird eine Umsetzung der Bildschirm-Darstellung in Graustufen durchgeführt. Dabei wird der Helligkeitseindruck des momentan angezeigten Bildes zugrundegelegt. Die Parameterwerte für die Hardcopy-Routine ermittelt das Programm automatisch. Jeder der 16 möglichen Farben ist über eine Tabelle der Wert 0 (weiß), 1 (hellgrau), 2 (dunkelgrau) und 3 (schwarz) zugeordnet. Aus diesem Grund empfiehlt es sich, das Bild auf einem monochromen Monitor oder einem Schwarzweiß-Fernseher zu betrachten (Farbmonitor: Farbintensität zurückdrehen).

### Programmhinweise und Druckeranpassung

Nachdem der »Hardmaker« im Speicher des C 64 steht, laden Sie das Drucker-Programm mit

```
LOAD "MULTIPRINT",8,1
```

Die Zeiger für Programmbeginn und -ende müssen mit folgenden POKE-Anweisungen im Direktmodus korrigiert werden:

```
POKE 43,1:POKE 44,8:POKE 45,30:POKE 46,23
```

In dieser Form arbeitet »Multiprint« mit Druckern zusammen, die am seriellen Port des C 64 angeschlossen und über Geräteadresse 4, Sekundäradresse 1 im Linearmodus betrieben werden. Die Geräteadresse ist in der Speicherstelle $16EA (5866) abgelegt. Die Drucker-Sekundäradresse steht in $16EB (5867).

Möchten Sie dagegen einen Drucker mit Parallelschnittstelle am Userport betreiben, geben Sie folgende Anweisung im Direktmodus ein:

```
POKE 5410,0
```

Dadurch wird der im Programm enthaltene Centronics-Druckertreiber aktiviert.

Wenn Sie noch ein übriges tun wollen, so sollten Sie die Voreinstellungen für die Multicolor-Farben ändern, damit Sie auch auf einem Farbmonitor eine Darstellung in Grauwerten erhalten. Sie müssen dazu lediglich zwei POKE-Befehle geben:

```
POKE 2088,1:POKE 2100,252
```

Jetzt können Sie die neue Version des Hardmaker speichern. (C. Kurts/M. Wilhelm/kn)




**Haben Sie als 64'er-Freak schon einmal neidisch auf Besitzer eines »echten« Personal-Computers geblickt? Dann wird es Zeit, ein wenig PC-Feeling beim C64 aufkommen zu lassen. »S-DOS« bietet den richtigen Einstieg in die Welt der »Großen«.**

Das Programm S-DOS besitzt eine dem Betriebssystem MS-DOS ähnliche Benutzeroberfläche. Das erleichtert PC-gewohnten Anwendern eine bequemere Nutzung des C64 und bietet dem Neuling die Chance, das Arbeiten mit der MS-DOS Oberfläche kennenzulernen. Teile des UNIX-Befehlssatzes wurden ebenfalls übernommen.

Falls Sie das erste Mal mit S-DOS arbeiten, laden Sie das Installationsprogramm von der beiliegenden Diskette mit:

```
LOAD "DOS/V2.1 INSTALL",8
```

und starten es mit RUN.

Nach einigen Sekunden erscheint die Aufforderung, eine Taste zu drücken. Legen Sie eine leere, formatierte Diskette ins Laufwerk und drücken Sie die Leertaste. Nach dem automatischen Speichervorgang wird S-DOS gestartet. Die gespeicherte Datei ist das fertig installierte Betriebssystem mit dem Filenamen »S-DOS-64 V2.1/TS«.

Um künftig mit diesem Programm zu arbeiten, muß es nicht mehr erneut installiert werden. Laden Sie das erzeugte Diskettenfile mit:

```
LOAD "S-DOS-64 V2.1/TS",8,1
```

Gestartet wird nicht mit RUN, sondern durch folgende Eingabe im Direktmodus:

```
SYS 64738
```

Ein Druck auf den Reset-Taster oder - bei Verwendung eines EPROMs - Einschalten des Computers erfüllt denselben Zweck. Der Basic-Interpreter des C64 arbeitet wie gewohnt, besitzt aber einen neuen Befehl:

```
DOS
```

Mit dieser Anweisung gelangt man in die eigentliche DOS-Benutzeroberfläche. Wer unbedingt ins Basic zurück will, muß nach Erscheinen des Prompts (A>) folgende Anweisung eingeben:

```
BASIC
```

Bei S-DOS wird Ihnen durch den »Prompt« signalisiert, daß Sie etwas eingeben können (vergleichbar mit der READY-Meldung in Basic). Der Prompt besteht im Normalfall aus dem Buchstaben »A« (für Laufwerk 8) und dem »Größer«-Zeichen (>). Das Aussehen des Prompts und das aktuelle Laufwerk können geändert werden. Davon später mehr. Wie beim MS-DOS-Kommandointerpreter der PCs (Zeileneditor), kann man sich bei S-DOS ebenfalls nicht mit dem Cursor frei auf dem gesamten Bildschirm bewegen. Eingabefehler werden wie gewohnt mit <DEL> gelöscht. Bei <CRSR rechts> wird ein Zeichen aus der letzten Eingabezeile geholt. Mehrmaliges Drücken dieser Taste bringt z.B. die gesamte Befehlseingabe auf den Bildschirm zurück.

Aus dem Betriebssystem MS-DOS wurden folgende Tastenfunktionen übernommen:

<CTRL BREAK>: Simuliert mit <CTRL RESTORE>. Dies bewirkt einen zwingenden Abbruch des Programms, auch im Basic-Modus. Ausnahme: siehe Break-Befehl.

<SHIFT PRTSCR>: Hier <SHIFT RESTORE>. Veranlaßt die Ausgabe des Textbildschirms auf einem Drucker. Die Taste funktioniert auch in Basic.

Die Laufwerksbezeichnungen sind »A« für Geräteadresse 8, »B« für Diskettenstation 9, »C« für 10 usw. Die Umschaltung auf ein anderes Laufwerk kann durch die Eingabe des entsprechenden Buchstabens erreicht werden. Falls die gewählte Floppystation nicht vorhanden ist, erscheint die Meldung »ERROR READING DRIVE«, die man mit <A> für »Abbruch« oder <I> für »Ignorieren« beantwortet. War das Laufwerk nicht eingeschaltet, können Sie die Taste <R> für einen neuen Versuch verwenden (»Repeat«).

Wird ein Befehl falsch eingegeben (fehlerhafte Parameter), erscheint der Fehlerhinweis »BAD COMMAND OR FILENAME«.

Mit der Eingabe eines Filenamens kann das Laden und Starten eines Programmes veranlaßt werden. Handelt es sich bei dem File um ein Basic-Programm, muß man es im Basic-Modus starten.

Viele Anweisungen sind wortgetreu aus dem Betriebssystem MS-DOS übernommen.

Hinweis: Der Vermerk (S) bedeutet, daß dieser Befehl eine Sicherheitsabfrage durchführt, ob Sie den Befehl wirklich durchführen wollen. (B) weist darauf hin, daß die Anweisung überwiegend für die Verwendung in einer Batchdatei (S-DOS-Anweisungen, die nacheinander abgearbeitet werden) gedacht ist.

Anstelle des Zeichens »/« bei der Parameterübergabe kann auch »-« verwendet werden.

### 1. Befehle zur Bildschirmsteuerung und allgemeine Anweisungen

**BASIC**: Springt ins Basic 2.0.
**CLS**: Löscht den Bildschirm.
**COLD**: Führt einen Kaltstart durch (S).
**CONFIG**: Gibt Gerätekonfiguration aus.
**EXIT**: Erzeugt einen Reset (S).
**HELP**: Gibt alle implementierten Befehle aus.
**MEMCLR**: Löscht den Speicher und veranlaßt den Kaltstart COLD (S).
**OFF**: Schaltet die Erweiterung ab. Die Bildschirmfarben und die Funktion <SHIFT RESTORE> bleiben erhalten.
**PROMPT**: Ändert das Eingabezeichen.
Parameter:
$G: Größerzeichen
$N: aktuelles Laufwerk (z.B. A)
$$: Zeichen »$«
$V: Versionsnummer
$←: Taste <RETURN>
$H: Taste <DEL>

Ist die Eingabe zu umfangreich, erscheint die Meldung: »NOT ENOUGH SPACE«.
**SETCOL**: Zeigt Bildschirmfarben an und läßt Änderungen zu.
Parameter:
- SETCOL Bx: Ändert Hintergrundfarbe.
- SETCOL Tx: Ändert Textfarbe. x=A-Z: A=schwarz, B=weiß, usw.

**VER**: Gibt die Versionsnummer des Programms aus.

## 2. Primäre Diskettenbefehle

**ADDR**: Nennt die Startadresse des zuletzt geladenen Files.

Parameter: ADDR filnam liest die Startadresse des Programms auf Diskette und gibt diese auf dem Bildschirm aus.

**COPY**: Kopiert Files. Nachdem COPY nach dem Source-File (Originalfile) gefragt hat, wird dieses geladen. Jetzt verlangt die Anweisung die Eingabe des Target-Files (Zielfile). Legen Sie eine andere Diskette in das Laufwerk. Wenn Sie dieselbe Diskette verwenden, muß bei Target-File ein anderer Filename gewählt werden. Vorsicht: Basic-Programme im Bereich von $0801 bis $7FFF werden überschrieben!

**DEL**: Löscht ein File von Diskette. Die Sicherheitsabfrage erfolgt nur, wenn »*« am Ende eines Filenamens vorkommt.

| Befehl | Funktion |
|---|---|
| ADDR | Adresse eines Files ausgeben |
| AGAIN | Tastenwiederholung |
| BAR | * senkrechte Menüauswahl (Balken) |
| BASIC | Rückkehr zum Basic 2.0 |
| BATLN | Adresse der Batchzeile ausgeben |
| BREAK | <RESTORE/CTRL> sperren/freigeben |
| CLS | Bildschirm löschen |
| COLD | Kaltstart |
| CONFIG | Konfiguration ausgeben |
| COPY | Files kopieren |
| CREATE | Batchdatei erzeugen |
| CRSR | * Cursor setzen |
| DEF | neuen Befehl definieren |
| DEL | Files löschen |
| DIR | Directory anzeigen |
| DISK | Disk-Befehl senden, Fehlerkanal auslesen |
| DO | Batchdatei ausführen |
| DUMP | File von Diskette listen |
| ECHO | * Text ausgeben |
| ENV | Environment-Variablen ausgeben |
| EXEASM | Assemblerroutine ausführen |
| EXEC | * File laden und ausführen |
| EXIT | SystemReset |
| FAST | File schnell laden |
| GET | Batchdatei laden |
| HELP | alle neuen Befehle ausgeben |
| IFJMP | * bedinger Sprung |
| INPUT | * String einlesen |
| INT | Interrupt sperren/freigeben |
| JMP | * unbedingter Sprung |
| KEY | Taste umdefinieren |
| LINE | Adresse einer Basic-Zeile ausgeben |
| LOAD | * File nur laden |
| MEMCLR | Speicher löschen, Kaltstart |
| OFF | Erweiterung abschalten |
| OUT | * Steuerzeichen ausgeben |
| PAUSE | * auf Taste warten |
| POP | * Rücksprungwert vom Stack holen |
| PROMPT | Bereitschaftszeichen ändern |
| REP | * Wiederholen, bis Wert = 0 |
| REPIN | * letzte Eingabe als nächsten Parameter |
| RUN | Basic-Befehl ausführen |
| SELECT | * waagerechte Menüauswahl (Balken) |
| SETBAT | Batchdatei: Defaultzeiger |
| SETCOL | Farben ändern |
| SETDEF | Adresse für Befehlssubst.Tabelle |
| SETEND | Ende der Batchdatei |
| SETENV | Adresse des Stacks für Batchdateien |
| SETINT | Interrupt setzen |
| SETKEY | Tastaturtabelle setzen |
| SETVAR | Basic-Variablen belegen |
| SPACE | Spacing an/aus |
| SWITCH | zwei Basic-Programme gleichzeitig laufen lassen |
| TYPE | Batchdatei listen |
| VAR | Basic-Variablen ausgeben |
| VER | Versionsnummer ausgeben |
| VOL | Diskettennamen ausgeben |
| WAIT | * auf Taste warten |
| WIN | * Fensterrahmen zeichnen |
| WINPUT | * String einlesen |

**Tabelle 1. »S-DOS«-Anweisungen in alphabetischer Reihenfolge. Das »*«-Zeichen bedeutet, daß dieser Befehl für eine Batchdatei geeignet ist.**

Beispiele:

```
DEL S-DOS
DEL FILE*
DEL FILE1, FILE2
```

**DIR**: Zeigt das Inhaltsverzeichnis der Diskette an.

Parameter:

/W: Kurzform

/P: wartet bei langen Directories nach jeder Bildschirmseite auf einen Tastendruck.

**DISK**: Gibt den Fehlerkanal aus. Wird ein Parameter übergeben, interpretiert ihn das Programm als Diskettenkommando. Ein Beispiel:

```
DISK I0
```

**DUMP**: Listet ein File von Diskette (Speicher bleibt unverändert). Dabei werden alle nicht druckbaren Zeichen (z.B. das Zeilenende-Zeichen »Carriage Return« <CR>) als reverse Characters im Bildschirmcode ausgegeben.

**EXEC**: Lädt Programm und führt es aus (B). Beispiel:

```
EXEC spaceinv
```

**FAST**: Lädt ein Programm sechsmal schneller in den Speicher. Parameter:

- FAST filnam (wie LOAD »Filename«,8,1)
- FAST $nnnn filnam (lädt die angegebene Datei an die Speicherstelle $nnnn)

**GET**: Lädt eine Batchdatei an die durch SETBAT definierte Adresse.

```
A>SETBAT $0400
A>GET batch1
```

lädt das Programm in den Bildschirmspeicher.

**LOAD**: lädt ein Programm (B).

```
LOAD spaceinv
```

**VOL**: gibt den Diskettennamen (Volume-Name) aus.

## 3. Konfigurationsbefehle und Schalter

**AGAIN**: Zeigt den Tastenwiederholungs-Status auf dem Bildschirm an. Parameter: »AGAIN ON« initialisiert die Wiederholfunktion, »AGAIN OFF« schaltet sie ab.

**BREAK**: Gibt den Breakstatus aus. Nach »BREAK ON« ist ein Abbruch mit <CTRL RESTORE> erlaubt, bei »BREAK OFF« ist die Tastenkombination wirkungslos. Die Parameter werden werden wie beim AGAIN-Befehl verwendet.

**ENV**: Zeigt alle Environmentvariablen an, die S-DOS als Adreßzeiger benutzt (SETBAT, SETINT, SETDEF, SETEND, SETENV, SETKEY und LDADDR). In LDADDR steht dieselbe Adresse wie in ADDR.

**INT**: Gibt Interruptstatus an. Bei INT ON ist ein Interrupt erlaubt, bei INT OFF nicht.

```
INT ON
```

springt an die durch SETINT vordefinierte Adresse, sofern der Interruptstatus vorher OFF war, ansonsten erfolgt keine Aktion. Bei INT OFF wird kein Interrupt zugelassen.

**SPACE**: Zeigt den Spacingstatus an. Ist das »Spacing« aktiviert (»SPACE ON«), werden führende Leerzeichen bei der Eingabe von Befehlen ignoriert.

**SETBAT**: Setzt den Defaultzeiger für Batchdateien (verwendet von DO, TYPE, GET, usw.) auf eine neue Adresse.

**SETDEF**: ändert Adresse für Befehlssubstitutionstabelle (Erklärung folgt). Danach sollte unbedingt

```
DEF /D
```

eingegeben werden, um einen Bereich zu löschen, der keine solche Tabelle enthält (Leertabelle). Anschließend können mit DEF neue Befehle definiert und erzeugt werden. Davon später mehr. Damit sich der C64 an die neuen Befehle erinnern kann, muß er sie irgendwo speichern. Dazu dient die Befehlssubstitutionstabelle (Substitution bedeutet so viel wie »Ersetzen«).

**SETEND**: Setzt den Wert für die Environmentvariable für SETEND (wird auch von CREATE verwendet, um das Ende einer gerade erstellten Batchdatei anzuzeigen).

**SETENV**: Ändert die Adresse des Stack für Batchdateien und den Inhalt des Prompt (Default: A>).

**SETINT**: Setzt die Environmentvariable SETINT auf eine neue Adresse. Beispiel:

```
SETINT $1000
```

In diesem Beispiel muß ab $1000 eine Assemblerroutine stehen, welche die IRQ-Vektoren $0314/$0315 initialisiert.

**SETKEY**: Ändert die Adresse für die Tastentabelle.

### 4. Ein- und Ausgabebefehle für Batchdateien

**ECHO**: Gibt einen Text auf dem Bildschirm aus.

Parameter:

ECHO <Text> gibt Text und am Ende der Zeile RETURN aus

ECHO-<Text> oder ECHO/<Text>: Ausgabe von Text (ohne RETURN)

ECHO alleinstehend gibt eine Leerzeile aus.

```
ECHO-ICH BEGRUESSE SIE
ECHO BEI DIESEM PROGRAMM
ECHO
ECHO GLEICH GEHT'S LOS!
```

**INPUT**: Liest einen String bis max. 30 Zeichen ein und hängt die Zeichenkette als Parameter an den folgenden Befehl.

**OUT**: Gibt Steuerzeichen aus. Zwei Beispiele:

OUT $05 setzt die Textfarbe auf Weiß.

OUT $FF gibt ein Pi aus.

**PAUSE**: Bringt einen festgelegten Text auf den Bildschirm und wartet auf einen Tastendruck.

**REPIN**: Hängt den zuletzt eingegebenen String von INPUT an den nächsten Befehl als Parameter.

**WAIT**: wartet auf Tastendruck (mit ECHO/ und WAIT kann PAUSE ersetzt werden).

### 5. Batchverarbeitungsbefehle

Batchdateien unter MS-DOS sind einfache Textdateien, die nahezu mit jedem Texteditor bearbeitet werden können. S-DOS stellt dazu entsprechende Befehle und einen Zeileneditor zur Verfügung. Die größten Unterschiede zu MS-DOS bestehen darin, daß S-DOS Unterbatchdateien ausführt (wie GOSUB in Basic).

**CREATE**: Erzeugt Batchdateien. Zuerst müssen die Befehle, jeweils in einer neuen Zeile, eingegeben werden. Um den Eingabevorgang abzuschließen, wird entweder der Klammeraffe »@« (für Exit ohne Speichern) oder der Pfeil oben »↑« (für Exit mit Speichern) am Zeilenanfang eingegeben.

```
DATEINAME
```

Parameter: /A für Append. Listet erneut die durch SETBAT definierte Batchdatei. An deren Ende lassen sich weitere Befehle eingeben.

**BATLN**: Gibt die Adresse der angegebenen Batchzeile aus (relativ zu SETBAT). Parameter:

- BATLN /S $xx: Die Variable SETBAT wird auf den errechneten Wert gesetzt. Damit kann mit CREATE eine beliebige Zeile editiert werden. Um jedoch die folgende Zeile nicht mit der Endmarke zu überschreiben, muß nach Druck auf <RETURN> anschließend <CTRL RESTORE> gedrückt werden.

**TYPE**: Listet eine bestehende Batchdatei auf. Default: SETBAT, d.h. TYPE listet die Batchdatei ab der durch SETBAT definierten Adresse. Parameter:

- TYPE $nnnn: Es wird eine Batchdatei ab Speicherstelle $nnnn ausgegeben.
- TYPE filnam: Die Datei »filnam« wird in den Speicher an die durch SETBAT definierte Adresse geladen.

**DO**: Führt eine Batchdatei aus. Default: SETBAT. Parameter:

- DO $nnnn: Eine Batchdatei ab $nnnn im Speicher wird ausgeführt
- DO filnam: »filnam« wird in den Speicher an die durch SETBAT definierte Adresse geladen.

DO kann mit dem Aufruf DO $nnnn bis zu 64 Verschachtelungen eingehen. Wird diese Zahl überschritten, so erscheint der Fehler »BATCH-OVERFLOW«.

Der Befehl SETBAT hat keinerlei Auswirkungen auf die gerade laufende Batchdatei, d.h. es können mit GET weitere Batchfiles in den Speicher geladen werden. Außerdem behandelt DO unbekannte Befehle als Kommentar. Deshalb lassen sich keine Filenamen im Programm angeben.

**POP**: Da DO (wie GOSUB) Werte auf den Stack legt, die nach Beendigung der Subroutine wieder geholt werden, kann kaum rekursiv programmiert werden.

```
A>CREATE
ECHO SCHLEIFE
DO
@
A>DO
```

führt nach dem 63. Aufruf zu »BATCH-OVERFLOW«, aber

```
A>CREATE
ECHO SCHLEIFE
POP
DO
@
A>DO
```

läuft und läuft und läuft...

Erklärung: POP holt einen Rücksprungwert vom Stapel. Mit dieser Technik können bereits überladene Batchdateien bei der Rückkehr übersprungen werden.

### 6. Schnittstelle zu Assembler

Die Befehle INT und SETINT für die Interruptbehandlung wurden bereits besprochen. Eine auf Diskette vorhandene Assemblerroutine kann mit LOAD, FAST, EXEC oder GET in den Speicher geladen werden.

**EXEASM**: Dieser Befehl führt eine Assemblerroutine aus. Beispiel:

```
EXEASM $C000
```

S-DOS überprüft, ob diese Routine zu einem Absturz führen könnte. Trifft dies zu, wird das Assemblerprogramm nicht gestartet, sondern eine Fehlermeldung ausgegeben.

### 7. Schnittstelle zu Basic

**RUN**: Führt einen Basic-Befehl unter S-DOS aus.

```
RUN PRINT A$
```

Es können auch Programmzeilen geändert und gelöscht werden.

```
RUN 10 PRINT A$
RUN 20
```

Der zweite Befehl löscht Zeile 20. Nach der Ausführung des Befehls wird ins DOS zurückgesprungen.

Mit der Anweisung

```
RUN RUN
```

können Basic-Programme gestartet werden. Über RUN und DEF lassen sich Basic-Subroutinen als neue DOS-Befehle verwenden.

**LINE**: Der Line-Befehl gibt die Adresse einer Basic-Zeile bzw. die der nächsthöheren Zeile aus, falls die gesuchte nicht vorhanden ist.

## Kurzinfo: S-DOS

**Programmart**: Betriebssystem/Benutzeroberfläche
**Laden**: LOAD "S-DOS-64 V2.1/TS",8,1
**Start**: Nach dem Laden eingeben: NEW:SYS 64738 und die RETURN-Taste drücken
**Steuerung**: Tastatur
**Besonderheiten**: auf EPROM oder im Speicher lauffähig, Fensterbefehle, ermöglicht relativ realistische Darstellung des PC-Systems »MS-DOS«. Kann auf anderen Disketten mit dem Installationsprogramm »DOS 2.1/INSTALL« erzeugt werden.
**Länge in Blocks**: 33 Blocks
**Programmautor**: Christian Dombacher

**SWITCH**: Umschalten von zwei Basic-Tasks (zwei Programme laufen gleichzeitig). Parameter: SWITCH $nnnn $mmmm. $nnnn und $mmmm geben die Startadressen der beiden Basic-Programme im Speicher an (siehe auch LINE).

```
SWITCH $0801 $0936
```

Parameter:

**/L** stellt eine Verbindung zwischen SWITCH und LINE dar. Beispiel: SWITCH /L 10 125 schaltet zwischen den Routinen, die in den Zeilen 10 und 125 beginnen, um.

**VAR**: Gibt Basic-Variablen aus. Beispiel: VAR A$ B$ C$. Bei Realzahl-Variablen hilft ein Trick: Man setzt ein Leerzeichen, anschließend zwei Anführungszeichen (") und ein SPACE.

**SETVAR**: Weist einer Basic-Variablen einen Wert zu.

```
SETVAR A 1.2 B X EF$ A$ F% 5
```

Dieser Befehl weist A den Wert 1.2, B den Wert von X, EF$ den Wert von A$ und der Variablen F% den Wert 5 zu.

### 8. Batchschleifen- und Sprungbefehle

**JMP**: Ein unbedingter Sprung.

JMP $00 springt an den Programmanfang (1. Zeile)

JMP $01 springt in die 2. Zeile

JMP funktioniert nur in Batchdateien.

**IFJMP**: Bedingter Sprung.

```
IFJMP $nn <Ausdruck>
```

$nn = Zeile, (siehe JMP);

<Ausdruck> = beliebiger, in Basic gültiger Ausdruck (wie IF, z.B. C AND B). IFJMP ist ebenfalls nur in Batchdateien anwendbar.

**REP**: Wiederholt so lange, bis der angegebene Wert »0« ist.

```
REP $nn $mm
```

$nn = Zeile, (siehe JMP);

$mm = Counterwert.

Da REP am Ende einer Schleife steht, sollte man den bereits ausgeführten Durchlauf berücksichtigen. Die beim REP-Befehl angegebene Zeilennummer darf nicht höher als die aktuelle Zeile sein.

### 9. Befehlssubstitution und Neubelegung der Tastatur

**KEY**: Definiert eine Taste um.

```
KEY $nn $mm
```

Der ASCII-Code $nn wird als $mm interpretiert.

```
KEY $5A $59
```

Z wird zu Y,

```
KEY $59 $5A (Y -> Z)
```

Y wird zu Z.

Diese Befehlsbeispiele vertauschen also die Tasten Z und Y. Parameter:

KEY /D: löscht Tastaturtabelle

KEY /Fx TEXT: belegt die Funktionstaste x (mögliche Werte sind 1 bis 8) mit TEXT. Bei Druck auf die entsprechende Funktionstaste wird dieser Befehl ausgeführt (es erfolgt keine Ausgabe auf dem Bildschirm).

```
KEY /F1 HELP
```

Nach Druck auf die Taste <F1> wird nun die HELP-Ausweisung ausgeführt.

Haben Sie mehr als 95 Tasten umdefiniert, erscheint die Fehlermeldung »KEYBUF-OVERFLOW«. Die Taste <RETURN> ($0D) kann nicht umdefiniert werden.

**DEF**: Erzeugt einen neuen Befehl. Das Programm fragt nach dem neuen Kommando und der zu ersetzenden Befehlssequenz. Dem neuen Befehl können keine Parameter übergeben werden. Parameter für DEF:

- DEF /D: löscht die Befehlstabelle
- DEF /S: zeigt alle neuen Befehle und ihre Funktion an
- DEF /L filnam: lädt Befehlstabelle »filnam« von Diskette
- DEF /W filnam: speichert die aktuelle Befehlstabelle auf die Floppy.

Zusätzlich können Kommandos wie DIR, PROMPT und HELP oder Laufwerksumschaltungen (A, B,...) mit neuen Funktionen belegt werden. Die alten Anweisungen sind noch vorhanden, sofern sie nicht belegt sind. Beliebig viele Kommandos lassen sich definieren. Die Anzahl wird lediglich durch den verfügbaren Speicherplatz begrenzt.

### 10. WINDOW-Befehle (Fenster)

**WIN**: Zeichnet einen Fensterrahmen auf den Bildschirm.

```
WIN $bbbb $xx $yy
```

$bbbb = Position der linken oberen Ecke im Bildschirmspeicher.

$xx = Länge +2 in X-Richtung (soll größer als 1 sein)

$yy = Länge in Y-Richtung (soll größer als 0 sein).

Ein Beispiel:

```
WIN $0459 $0e $03
```

Zeichnet einen Fensterrahmen mit der linken oberen Bildschirmposition 1113, 14 Spalten Länge (X-Richtung) und 3 Zeilen Breite (Y-Richtung).

Der Inhalt des Fensters muß direkt hinter WIN im Batchprogramm stehen (Anzahl Zeilen = Länge Y-Richtung). Der Befehl WIN funktioniert nur in Batchdateien.

**BAR**: Ermöglicht eine senkrechte Auswahl mit Hilfe eines invertierten Balkens.

```
BAR $bbbb $xx $yy
```

(Belegung wie bei WIN-Befehl)

BAR übergibt dem nächsten Befehl den Parameter A für die 1. Zeile, B für die 2. Zeile, usw. Um eine gezielte Abfrage zu erreichen, sollte der Aufruf SETVAR XX$ direkt hinter BAR stehen. Danach enthält die gewählte Variable das Ergebnis mit führendem Leerzeichen. Mit IFJMP kann gezielt verzweigt werden. IFJMP-Beispiele finden Sie in der Batchdatei »SHORT-DEMO« auf der beiliegenden Diskette.

**WINPUT**: Liest einen String ein.

```
WINPUT $bbbb $xx.
```

Belegung wie bei WIN-Befehl. WINPUT gibt wie INPUT den eingelesenen String an den folgenden Befehl weiter (hier funktioniert REPIN ebenfalls).

```
WINPUT $0485 $10
```

**SELECT**: Ermöglicht eine waagrechte Auswahl mit Hilfe eines invertierten Balkens.

Beispiel: SELECT $zz $ss $aa $gg

$zz = linkes Element Zeilenposition

$ss = linkes Element Spaltenposition

$gg = gesamte Anzahl aller auswählbaren Werte

$aa = Anfangswert, auf diesem Element steht der invertierte Balken nach dem Aufruf. Das Zeichen links von jedem Element muß ein Leerzeichen sein (deshalb funktioniert die Position zz=$00 ss=$00 nicht einwandfrei). Der Text sollte vor dem Aufruf auf dem Bildschirm stehen. SELECT liefert das gleiche Ergebnis wie BAR.

**CRSR**: Positioniert den Cursor.

```
CRSR $zz $ss.
```

Belegung entspricht SELECT-Befehl. Dieser Befehl sollte zur Positionierung und Ausgabe eines Strings vor dem SELECT-Befehl verwendet werden.

```
CRSR $0A $01
```

Der Cursor erscheint in der ersten Spalte der 10. Zeile.

Auf der beiliegenden Diskette finden Sie unter »SHORT-DEMO« ein Programmbeispiel für eine Batchdatei. Laden Sie das File unter S-DOS mit der Anweisung

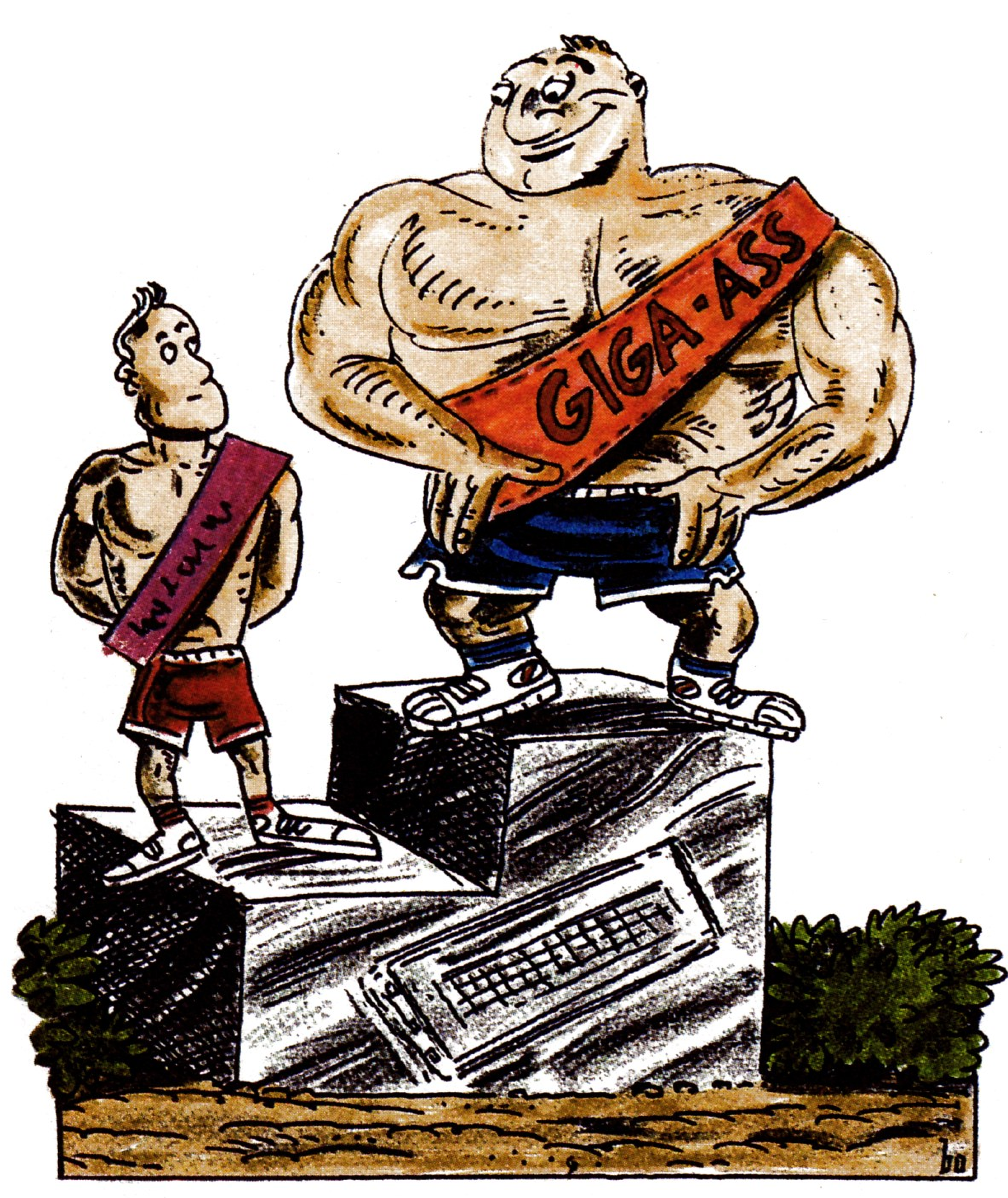
```
A>TYPE SHORT-DEMO
```

und analysieren Sie zur Übung die auf dem Bildschirm angezeigten Programmzeilen. Nachdem der Prompt »A>« erschienen ist, läßt sich das Programm mit »DO« starten.

Zur besseren Übersicht sind in Tabelle 1 alle S-DOS-Befehle mit kurzer Erklärung aufgelistet.

Zu Beginn sollten Sie erst einmal ausgiebig mit S-DOS »spielen«. S-DOS bietet Ihnen einen kleinen Einblick in die Welt der PCs. Wir wünschen Ihnen dabei viel Vergnügen!

(Christian Dombacher/Nikolaus M. Heusler/bl)




# Assembler der Superlative

**Ein ideales Werkzeug für Programmier-Freaks ist »Giga-Ass«. Durch eine optimale Speicherausnutzung läßt dieser komfortable Makro-Assembler keine Wünsche offen.**

Über 20 Editor-Befehle – damit läßt sich Assembler fast so einfach wie Basic programmieren. »Giga Ass« belegt den ROM-Bereich von $8000 bis $9FFF und wird wahlweise im RAM betrieben oder über ein EPROM-Modul eingeblendet. In Tabelle 1 finden Sie eine Übersicht mit den wichtigsten Merkmalen von Giga-Ass.

Der Assembler Giga-Ass belegt den ROM-Bereich von $8000 bis $9FFF und wird wahlweise im RAM betrieben oder über ein EPROM-Modul eingeblendet. In Tabelle 1 (Seite 16) finden Sie eine Übersicht mit den wichtigsten Merkmalen von Giga-Ass.

a) Sie verwenden das Original-Betriebssystem des C64. Dann geben Sie ein:

```
LOAD " GIGA-ASS" ,8,1, <RETURN>
```

Nach dem Laden SYS 64738 und <RETURN> eingeben. Es erscheint die Einschaltmeldung des Assemblers.

Verfügt Ihr C64 über einen Reset-Knopf, genügt es, diesen nach dem Laden zu drücken. Da Giga-Ass durch die »CBM80«-Kennung ein Modul simuliert, wird der Assembler bei einem Reset sofort gestartet.

Giga-Ass kann auch direkt auf ein 2764-EPROM gebrannt werden. Die EPROM-Platine muß dann auf den Speicherbereich von $8000 bis $9FFF adressiert werden.

Als Quelltextspeicher stehen 30 KByte RAM von $0800 bis $7FFF zur Verfügung. Als Speicher für die Symboltabelle (Labels, Makronamen) dient der unter dem Basic-Interpreter befindliche RAM-Bereich von $A000 bis $BFFF. Dieser nimmt 1170 Symbole auf, da aufgrund des besonders kompakten Formats jedes Symbol nur 7 Byte belegt. Der Speicher $C000 bis $CFFF bleibt für einen Monitor frei. Giga-Ass und der SMON können sich also gleichzeitig im Speicher befinden.

Der Quelltext wird wie ein Basic-Programm mit dem zeilenorientierten Editor von Basic eingegeben und in einem kompakten Format im Speicher abgelegt. Sämtliche Assemblerbefehle werden in Form von »Token« kodiert, das heißt, jeder Assembler-Befehl braucht nur 1 Byte Speicherplatz. Außerdem lassen sich alle Befehle auch abgekürzt eingeben; in den meisten Fällen genügt dafür eine einzige Taste. Nach Eingabe einer Zeile wird diese sofort auf dem Bildschirm in Klartext und formatiert angezeigt.

Zum eigentlichen Assemblierungs-Vorgang muß sich Quelltext im Speicher befinden. Das Objektprogramm wird entweder auf Diskette oder in den Speicher geschrieben, wobei sich durch eine zusätzliche Code-Adresse der Speicherbereich, in dem das Programm abgelegt wird, frei wählen läßt. Die Erzeugung des Objektprogramms läßt sich aber auch ganz unterdrücken, damit nur ein reiner Syntax-Check vorgenommen wird.

Während des Assemblierens kann man ein Listing des Quelltextes auf den Drucker oder auf Diskette ausgeben. Eine Bildschirmausgabe ist bei 40 Zeichen pro Zeile weniger sinnvoll.

Nach Abschluß der Assemblierung wird die Symboltabelle wahlweise auf dem Bildschirm, auf Diskette oder Drucker ausgegeben. Alle Symbole werden mit Namen und Symbolwert aufgelistet. Außerdem steht bei jedem Symbol die Zeilennummer der Zeile, in der es verwendet wurde, und der Symboltyp.

GIGA-ASS unterscheidet drei Typen von Symbolen:

1. Makronamen
2. Globale Label
3. Lokale Label

Makros können an jeder Stelle des Quelltextes definiert und verwendet werden. Ein Makroaufruf unterscheidet sich in nichts von einem Assemblerbefehl: Es wird nur der Makroname eingegeben. Eventuelle Parameter werden von dem Makro-Namen durch ein Leerzeichen (<SPACE>) und untereinander durch Kommata getrennt.

### Die Eingabe von Qelltextzeilen

Eine Quelltextzeile wird wie eine Basic-Zeile eingegeben. Damit der Editor die einzelnen Elemente einer Quelltextzeile (Zeilennummer, Label, Assembler-Befehl und Kommentar) voneinander unterscheidet, müssen Sie einige Regeln beachten:

**Kurzinfo: Giga-Ass**

**Programmart:** Makro-Assembler zum Erstellen von Maschinenprogrammen
**Laden:** LOAD "GIGA-ASS",8,1
**Starten:** nach dem Laden SYS64738 <RETURN> eingeben
**Steuerung:** Tastatur
**Benötigte Blocks:** 33 Blocks und 74 Blocks Zusatzprogramme.
**Programmautor:** Thomas Dachsel

1. Jede Quelltextzeile beginnt mit einer eindeutigen Zeilennummer. Diese wird entweder direkt eingegeben oder mit dem Auto-Befehl vorgegeben.

2. Unmittelbar nach den Zeilennummern folgen die Label. Zwischen Zeilennummer und Label darf kein Leerzeichen stehen.

3. Ein auf ein Label folgender Assemblerbefehl muß vom Label durch ein Leerzeichen getrennt werden.

4. Enthält eine Zeile kein Label, so beginnt sie mit einem Leerzeichen nach der Zeilennummer, welches unbedingt eingegeben werden muß.

5. Kommentare werden am Ende der Zeile angefügt und vom Rest der Zeile durch ein Semikolon getrennt. Reine Kommentarzeilen haben als erstes Zeichen in der Zeile ein Semikolon.

6. Enthält eine Quelltextzeile nur ein Label und keine Assembleranweisung, so darf auf das Label kein Kommentar folgen!

**Die Editor-Befehle**

Da der Assembler-Editor eine Erweiterung des Basic-Editors darstellt, sind die normalen Basic-Befehle wie OPEN, CLOSE, CMD, PRINT #, POKE, PEEK und SYS wie gewohnt verwendbar. Nicht erlaubt sind alle Basic-Befehle, die Variablen anlegen, also LET, FOR, NEXT, DIM etc.

Die neu hinzugekommenen Editor-Befehle sind im Gegensatz zu den Basic-Befehlen nur ein Zeichen lang. Darauf ist bei Überschneidungen zu achten. Zum Beispiel heißt der Editor-Befehl zum Laden von Quelltexten »L«, während der Basic-Befehl bekanntlich LOAD lautet.

Die Unterscheidung zwischen Basic- und den neuen Befehlen wird vorgenommen, indem geprüft wird, ob das zweite Zeichen einer Eingabe ein (eventuell geSHIFTeter) Buchstabe ist. Dies ist bei Basic-Befehlen immer der Fall, bei den anderen Befehlen ist das zweite Zeichen eine Zahl oder ein Hochkomma.

**A (Auto)**

dient zur automatischen Vorgabe der Zeilennummern bei der Eingabe neuer Quelltextzeilen. Syntax:

A Zeilennummer, Schrittweite

Zum Beispiel gibt A 100,10 die Zeilennummern ab 100 in Zehnerschritten vor (100, 110, 120, 130 etc.). Die Numerierung wird abgebrochen, indem man mit <SHIFT RETURN> den Cursor in eine leere Zeile bewegt und <RETURN> drückt. Gibt man im Auto-Modus direkt hinter einer Zeilennummer <RETURN> ein, so wird diese übersprungen und sofort die nächste Zeilennummer ausgegeben. Durch Drücken von <RETURN> können so größere Bereiche übersprungen werden. Befinden Sie sich im Auto-Modus, so lassen sich bereits eingegebene Zeilen ändern, ohne den Auto-Modus zu verlassen. Voraussetzung ist jedoch, daß die betreffenden Zeilen sich noch auf dem Bildschirm befinden. Man setzt die Numerierung fort, indem man in der Quelltextzeile mit derjenigen Zeilennummer, die zuletzt vorgegeben wurde, <RETURN> drückt.

Die Numerierung kann nach Abbruch automatisch an der alten Stelle wieder aufgenommen werden durch Drücken von <A> und <RETURN>.

**Steckbrief Giga-Ass**

- 30 KByte Quelltextspeicher
- Quelltexte kompatibel zu Hypra-Ass (mit Konvertierprogramm)
- Komfortable Funktionstastenbelegung
- Kompakter Quellcode durch Token-Verarbeitung
- Programm liegt in brennfähiger Version vor (für EPROM-Module)
- Modulstart integriert (CBM80)
- Resetfest
- Platz für Monitor ab $C000
- OLD-Funktion
- Assemblieren direkt auf Diskette
- Formatierte Ausgabe von Symboltabellen
- Merge-Funktion für Quelltexte
- Basic-Befehle im Direktmodus weiterhin nutzbar
- Bis zu 1170 Symbole (Label, Makronamen, etc.) verwendbar

**Tabelle 1. »Giga-Ass« auf einen Blick**

| Token | Befehl | Taste 1 | Taste 2 |
|---|---|---|---|
| $A0 | .CALL | <SHIFT SPACE> | |
| $A1 | .MACRO | <CBM K> | <SHIFT A> |
| $A2 | .ENDMACRO | <CBM I> | <SHIFT B> |
| $A3 | .GLOBAL | <CBM T> | <SHIFT C> |
| $A4 | .EQUATE | <CBM @> | <SHIFT D> |
| $A5 | .BYTE | <CBM G> | <SHIFT E> |
| $A6 | .WORD | <CBM +> | <SHIFT F> |
| $A7 | .DS | <CBM M> | <SHIFT G> |
| $A8 | .EXT | <CBM £> | <SHIFT H> |
| $A9 | .OBJECT | <SHIFT £> | <SHIFT I> |
| $AA | .BASE | <CBM N> | <SHIFT J> |
| $AB | .CODE | <CBM Q> | <SHIFT K> |
| $AC | .ON | <CBM D> | <SHIFT L> |
| $AD | .GOTO | <CBM Z> | <SHIFT M> |
| $AE | .IF | <CBM S> | <SHIFT N> |
| $AF | .ELSE | <CBM P> | <SHIFT O> |
| $F0 | .ENDIF | <CBM A> | <SHIFT P> |
| $F1 | .SYMBOLS | <CBM E> | <SHIFT Q> |
| $F2 | .LISTING | <CBM R> | <SHIFT R> |
| $F3 | .END | <CBM W> | <SHIFT S> |
| $F4 | .STOP | <CBM H> | <SHIFT T> |
| $F5 | .PAGE | <CBM J> | <SHIFT U> |
| $F6 | .NOCODE | <CBM L> | <SHIFT V> |
| $F7 | .START | <CBM Y> | <SHIFT W> |
| $F8 | .NOEXP | <CBM U> | <SHIFT X> |

**Tabelle 2. Token-Tabelle für Pseudo-Befehle**

**B (Speicheranzeige)** <F2>

zeigt den gesamten für den Quelltext zur Verfügung stehenden und den davon bereits benutzten Speicherbereich in KByte an. Zusätzlich wird die Anzahl der belegten und noch freien Bytes ausgegeben. Der B-Befehl wird auch durch <F2> aufgerufen.

**C (Kaltstart)**

bewirkt einen Neustart des Assemblers. Die Speicherkonfiguration wird nicht verändert. Es handelt sich nicht um einen Reset, sondern um eine Re-Initialisierung. Mit <O> (Old) und <RETURN> wird der Quelltext zurückgeholt.

**D (Delete)**

dient zum Löschen von Quelltextzeilen. Syntax:

D Zeilennummer 1 - Zeilennummer 2

Lassen Sie Zeilennummer 1 oder Zeilennummer 2 fortfallen, so wird von Quelltextanfang beziehungsweise bis Quelltextende gelöscht. Zum Beispiel bewirkt D -100, daß alle Zeilen bis einschließlich Zeile 100 gelöscht werden, und D 1000 -, daß alle Zeilen ab Zeile 100 einschließlich gelöscht werden.

D Zeilennummer löscht eine einzelne Zeile. Alternativ dazu gibt man die Zeilennummer allein und danach <RETURN> ein.

»D« ohne Argument löscht den gesamten Quelltext und entspricht damit dem weiterhin verwendbaren Basic-Befehl NEW.

Vor Ausührung des Löschbefehls erfolgt mit »ARE YOU SURE?« stets eine Sicherheitsabfrage. Nach Bestätigung mit <Y> wird die Operation ausgeführt; jede andere Taste führt zum Abbruch des Befehls.

**E (List)** <F1>

dient zum Ausgeben von Quelltext. Syntax:

E Zeilennnummer 1 - Zeilennummer 2

Wie beim D-Befehl können Zeilenbereiche, einzelne Zeilen oder der gesamte Quelltext geLISTet werden. Der E-Befehl ohne Argumente wird auch durch <F1> aufgerufen. Statt des E-Befehls ist auch das Basic-Befehlswort LIST verwendbar.

Mit <SPACE> wird das LISTen angehalten. Durch erneutes Drücken von <SPACE> fährt das LISTen fort. Mit <RUN/STOP> wird der Vorgang abgebrochen.

Eine Ausgabe des Quelltextes auf einen Drucker erreicht man durch:

OPEN1,4,7 : CMD1 <RETURN>
E <RETURN>

**F (Find)**
ermöglicht das Auffinden von Zeichenfolgen im Quelltext. Syntax:

F P,″Zeichenfolge″

| Token | Befehl | Taste | Token | Befehl | Taste |
|---|---|---|---|---|---|
| $C0 | CPX | <SHIFT *> | $DC | TXS | <CBM -> |
| $C1 | CPY | <SHIFT A> | $DD | PHP | <SHIFT -> |
| $C2 | LDX | <SHIFT B> | $DE | PLP | <SHIFT ↑> |
| $C3 | LDY | <SHIFT C> | $DF | PHA | <CBM *> |
| $C4 | CMP | <SHIFT D> | $E0 | PLA | <SHIFT SPACE> |
| $C5 | ADC | <SHIFT E> | $E1 | BRK | <CBM K> |
| $C6 | AND | <SHIFT F> | $E2 | RTI | <CBM I> |
| $C7 | DEC | <SHIFT G> | $E3 | RTS | <CBM T> |
| $C8 | EOR | <SHIFT H> | $E4 | NOP | <CBM @> |
| $C9 | INC | <SHIFT I> | $E5 | CLC | <CBM G> |
| $CA | LDA | <SHIFT J> | $E6 | SEC | <CBM +> |
| $CB | ASL | <SHIFT K> | $E7 | CLI | <CBM M> |
| $CC | BIT | <SHIFT L> | $E8 | SEI | <CBM £> |
| $CD | LSR | <SHIFT M> | $E9 | CLV | <SHIFT £> |
| $CE | ORA | <SHIFT N> | $EA | CLD | <CBM N> |
| $CF | ROL | <SHIFT O> | $EB | SED | <CBM Q> |
| $D0 | ROR | <SHIFT P> | $EC | DEY | <CBM D> |
| $D1 | SBC | <SHIFT Q> | $ED | INY | <CBM Z> |
| $D2 | STA | <SHIFT R> | $EE | DEX | <CBM S> |
| $D3 | STX | <SHIFT S> | $EF | INX | <CBM P> |
| $D4 | STY | <SHIFT T> | $F0 | BPL | <CBM A> |
| $D5 | JMP | <SHIFT U> | $F1 | BMI | <CBM E> |
| $D6 | JSR | <SHIFT V> | $F2 | BVC | <CBM R> |
| $D7 | TXA | <SHIFT W> | $F3 | BVS | <CBM W> |
| $D8 | TAX | <SHIFT X> | $F4 | BCC | <CBM H> |
| $D9 | TYA | <SHIFT Y> | $F5 | BCS | <CBM J> |
| $DA | TAY | <SHIFT Z> | $F6 | BNE | <CBM L> |
| $DB | TSX | <SHIFT +> | $F7 | BEQ | <CBM Y> |

**Tabelle 3. Token-Tabelle für Assembler-Befehle**

Ein Beispiel: F0,″LDA″ findet jedes Auftreten des Assembler-Befehls LDA im Quelltext und listet die entsprechenden Zeilen auf. Diese Auflistung wird durch Drücken der SPACE-Taste angehalten und danach durch nochmaliges Drücken von <SPACE> fortgesetzt. Die Auflistung brechen Sie dadurch ab, daß Sie sie mit <SPACE> anhalten und danach <RUN/STOP> drücken.

Der Parameter »P« (0 bis 15) dient zur Angabe der Page, also eines Bereiches im Arbeitsspeicher (siehe hierzu auch die entsprechende Erläuterung zum P-Befehl).

Das Fragezeichen kann im Suchbegriff als Joker verwendet werden: Es steht für jedes beliebige Zeichen.

Vor Ausführung des F-Befehls wird automatisch die Wandlung von Assemblerbefehlen in Token vorgenommen. Das hat zur Folge, daß Zeichenfolgen, die Assembler-Befehlswörter eingebettet haben, so nicht gefunden werden. Zum Beispiel wird das Wort »STATUS« deswegen nicht gefunden, weil in diesem Wort der Befehl STA enthalten ist. Diese Eigenschaft läßt sich umgehen, indem man einen Buchstaben von STA durch den Joker ersetzt und beispielsweise nach »ST?TUS« sucht.

Beim Packen der Quelltextzeilen wird jedes überflüssige Leerzeichen entfernt. Sucht man beispielsweise nach »JSR LABEL«, so hat man deswegen Pech, weil Leerzeichen zwischen Assemblerbefehlen und Operanden stets entfernt worden sind. »JSRLABEL« als Suchwort bringt dagegen Erfolg.

Eine Wandlung von Pseudo-Befehlstexten in die entsprechenden Token wird hier nicht vorgenommen. Beim F-Befehl muß man daher die Token direkt eingeben. Statt nach ».MACRO« zu senden, muß man also <CBM>, <K> oder alternativ <SHIFT A> eingeben. In der Tabelle 2 und 3 finden Sie die Tastenkombinationen für alle Befehle.

**G (obere Speichergrenze)**
setzt die obere Speichergrenze für den Assembler. Syntax:

G (Adresse als Hex-Zahl)

Es ist die höchste Adresse anzugeben, die noch vom Assembler für das Speichern von Quelltext verwendet werden darf. Die Adresse darf nicht größer oder gleich $8000 sein. Voreinstellung ist G $7FFF; damit stehen die vollen 30 KByte für den Quelltext zur Verfügung. Durch G $5FFF würde beispielsweise der Speicherbereich von $6000 bis $7FFF für Objektcodes reserviert und gegen Überschreiben durch Quelltext geschützt.

**I (Inhaltsverzeichnis)** <F7>
gibt das Inhaltsverzeichnis der sich gerade im Laufwerk mit der Nummer 8 befindlichen Diskette aus. Durch Drücken der SPACE-Taste kann die Ausgabe abgebrochen werden. Dieser Befehl wird auch durch <F7> aufgerufen.

**L (Load)** <F5>
lädt einen Quelltext in den Speicher. Syntax:

L″Filename″, Gerätenummer

Der Quelltext wird immer an die Adresse geladen, die in $2B/$2C (43/44) im Low-/High-Byte-Format steht. Dies wird im Regelfall der Basic-Start $0801 (2049 dez.) sein. Dieser Befehl wird auch durch <F5> aufgerufen. Wählen Sie dagegen den Befehl

LOAD ″Filename″,Gerätenummer

so wird die Angabe der Sekundäradresse 1 automatisch angenommen und das Programm absolut - also an die auf Diskette angegebene Startadresse - geladen. Eine weitere interessante Möglichkeit, Quelltexte aus dem angezeigten Directory zu laden, ist die Tastenkombination <SHIFT RUN/STOP>. Es wird nach dem Laden sofort mit der Assemblierung begonnen. Bei Verwendung von Speeddos, wozu eine bedingte Übereinstimmung der Funktionstastenbelegung existiert, kann die Angabe der Laufwerksnummer sowie der Doppelpunkt hinter dem Filenamen beim Laden aus dem Directory entfallen.

**M (Merge)**
lädt einen Quelltext hinter den bereits im Speicher befindlichen. Syntax:

M ″Filename″

Dieser Befehl wird dazu benutzt, um mehrere Quelltexte, die einzeln eingegeben wurden, von Diskette nachzuladen und zu verketten. Unter besonderen Bedingungen ist es realisierbar, mehrere unterschiedliche Quelltexte nacheinander mit dem M-Befehl einzuladen und dann in einem Durchgang zu assemblieren. Zu diesen Bedingungen zählt, daß die Speicherbereiche für die Objektcodes sich nicht überschneiden, falls direkt in den Speicher assembliert wird, und daß die Symbole in einem Quelltext nicht in einem weiteren Quelltext anders definiert werden.

**N (Number)**
ermöglicht ein neues Durchnumerieren von Quelltextzeilen. Syntax:

N P, Startnummer, Schrittweite

Zum Beispiel numeriert N0, 100, 10 den gesamten Quelltext neu, beginnend mit den Zeilennummern 100, 110, 120

# So finden Sie die Programme auf der Diskette

## DISKETTE SEITE 1

```
0    "64'ER SONDERH.53" S1 2A
0    "----------------" USR
0    "----------------" USR
0    "-----GRAFIK-----" USR
0    "----------------" USR
0    "----------------" USR
42   "GRAFIK 2001"      PRG     Seite 4
5    "GRA1"             PRG
38   "TESTDEMO"         PRG
1    "TESTSPRITE6"      PRG
11   "STATISTIK"        PRG
1    "DEMO.RECHT"       PRG
1    "DEMO.RORECHT"     PRG
1    "DEMO.LINE"        PRG
1    "DEMO.ELLIPSE"     PRG
1    "DEMO.ELLIPSE2"    PRG
1    "DEMO.CIRCLE"      PRG
```

```
2    "DEMO.CIRCLE2"     PRG
2    "DEMO.TEXT"        PRG
0    "----------------" USR
14   "HARDMAKER"        PRG     Seite 8
4    "MULTIPRINT"       PRG
0    "----------------" USR
0    "----------------" USR
0    "----ANWENDUNG---" USR
0    "----------------" USR
0    "----------------" USR
27   "DOS V2.1/INSTALL" PRG     Seite 11
4    "SHORT-DEMO"       PRG
33   "S-DOS-64 V2.1/TS" PRG
2    "LIST COMMANDS"    PRG
0    "----------------" USR
0    "----------------" USR
```

```
0    "----ASSEMBLER---" USR
0    "----------------" USR
0    "----------------" USR
33   "GIGA-ASS"         PRG     Seite 15
4    "SOURCE INIT 31 K" PRG
1    "INIT 31 K"        PRG
56   "HYPRA-KONV.SRC"   PRG
13   "HYPRA-CONVERT"    PRG
0    "----------------" USR
1    "PROMON-LADER"     PRG     Seite 28
33   "PROMON 64"        PRG
0    "----------------" USR
0    "----DISKETTE----" USR
0    "---BEIDSEITIG---" USR
0    "----BESPIELT----" USR
0    "----------------" USR
332 BLOCKS FREE.
```

## DISKETTE SEITE 2

```
0    "64'ER SONDERH.53" S2 2A
0    "----------------" USR
0    "----------------" USR
0    "------DFUE------" USR
0    "----------------" USR
0    "----------------" USR
47   "PROTERM V6.0"     PRG     Seite 32
1    "PRO.KEYS"         SEQ
1    "PRO.TEL"          SEQ
0    "----------------" USR
0    "----------------" USR
0    "------SOUND-----" USR
0    "----------------" USR
```

```
0    "----------------" USR
3    "LISTING 1"        PRG     Seite 35
3    "LISTING 2"        PRG
5    "LISTING 3"        PRG
8    "LISTING 4"        PRG
0    "----------------" USR
51   "SOUND-MONITOR"    PRG     Seite 42
1    "MUSIC-SWITCH"     PRG
3    "MUSIC CRUNCHER"   PRG
45   "AXEL F."          PRG
93   "DANCE AT NIGHT"   PRG
0    "----------------" USR
0    "----------------" USR
```

```
0    "-----SPIELE-----" USR
0    "----------------" USR
0    "----------------" USR
44   "CRILLION"         PRG     Seite 48
1    "TOP 8"            PRG
0    "----------------" USR
8    "BULLY"            PRG     Seite 47
0    "----------------" USR
0    "----------------" USR
0    "------ENDE------" USR
0    "----------------" USR
0    "----------------" USR
350 BLOCKS FREE.
```

# WICHTIGE HINWEISE

## zur beiliegenden Diskette:

**Aus den Erfahrungen der bisherigen Sonderhefte mit Diskette wollen wir ein paar sinnvolle Tips an Sie weitergeben:**

**1** Bevor Sie mit den Programmen auf der Diskette arbeiten, sollten Sie unbedingt eine Sicherheitskopie der Diskette anlegen. Verwenden Sie dazu ein beliebiges Backup-Programm, das eine komplette Diskettenseite kopiert.

**2** Auf der Originaldiskette ist wegen der umfangreichen Programme nur wenig Speicherplatz frei. Dies führt bei den Anwendungen, die Daten auf die Diskette speichern, zu Speicherplatzproblemen. Kopieren Sie daher das Programm, mit dem Sie arbeiten wollen, mit einem File-Copy-Programm auf eine leere, formatierte Diskette und nutzen Sie diese als Arbeitsdiskette.

**3** Die Rückseite der Originaldiskette ist schreibgeschützt. Wenn Sie auf dieser Seite speichern wollen, müssen Sie vorher mit einem Diskettenlocher eine Kerbe für die Rückseite der Diskette machen, um den Schreibschutz zu entfernen. Probleme lassen sich von vornherein vermeiden, wenn Sie die Hinweise unter Punkt 2 beachten.

# ALLE PROGRAMME
## aus diesem Heft




# HIER

**SONDERHEFT 54**

# VORSCHAU




Absolute Top-Spiele, vom Adventure über heiße Action-Games bis zum Rollenspiel – all das finden Sie im Sonderheft 54 und der beiliegenden Diskette.

- »Crillion II«, der Nachfolger des bekannten »Joystick-Killers«: noch spannender, rasanter und atemberaubender als die alte Version.
- »Samurai« bietet Ihnen die klassische Kampfsportart der geheimnisvollen »Ninjas« in professioneller Manier.
- »Kini Ludwig« ist ein Rollenspiel, das nicht nur die Lachmuskeln, sondern vor allem die »kleinen grauen Zellen« gehörig strapaziert.
- Das gab es noch nie: Sechs Spieler spielen gleichzeitig gegeneinander! »Ultimate Tron II« ist die irrste Verfolgungsjagd, die es je gab.

Nie mehr »Game over«, auch bei kommerziellen Spielen? Ein ausgefuchster Freak plaudert »Cracker«-Geheimnisse aus.

Brandaktuell und rechtzeitig zur Fußball-WM 1990 in Italien kommt »Italia '90«. Ein Programm, das Ihnen alle Spielpaarungen, Ergebnisse und Plazierungen Sekunden nach Spielende zeigt.

**Das Sonderheft 54 liegt ab 25.5.90 an Ihrem Kiosk.**


## Impressum

***Herausgeber:*** *Carl-Franz von Quadt, Otmar Weber*

***Redaktionsdirektor:*** *Richard Kerler*
***Chefredakteur:*** *Wolfram Höfler (hö)*
***Stellv. Chefredakteur:*** *Gottfried Knechtel (kn) – verantwortlich für den redaktionellen Teil*
***Chef vom Dienst:*** *Susanne Kirmaier*
***Redaktion:*** *Andreas Greil (ag), Harald Beiler (bl), Herbert Großer (gr)*
***Mitarbeiter dieser Ausgabe:*** *Nikolaus Heusler*
***Redaktionsassistenz:*** *Brigitte Bobenstetter, Sylvia Derenthal, Helga Weber (202)*
***Telefax:*** *089/4613-778.* ***Hotline*** *(640): Montag bis Donnerstag 16 bis 17 Uhr, Freitag 11 bis 12 Uhr*
*Alle Artikel sind mit dem Kurzzeichen des Redakteurs und/oder mit dem Namen des Autors/Mitarbeiters gekennzeichnet*

***Manuskripteinsendungen:*** *Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.*

***Art-director:*** *Friedemann Porscha*
***Titelgestaltung:*** *Wolfgang Berns*
***Layout:*** *Marian Schwarz*
***Bildredaktion:*** *Janos Feitser (Ltg.), Sabine Tennstaedt; Roland Müller (Fotografie); Ewald Standke, Norbert Raab (Spritzgrafik); Werner Nienstedt (Computergrafik)*

***Anzeigendirektion:*** *Ralph Peter Rauchfuss*
***Anzeigenleitung:*** *Philip Schiede (399) – verantwortlich für die Anzeigen*
***Telefax:*** *089/4613-775*

***Anzeigenverwaltung und Disposition:*** *Monika Burseg (147)*

***Auslandsrepräsentation:***
***Auslandsniederlassungen:***
***Schweiz:*** *Markt&Technik Vertriebs AG, Kollerstr. 37, CH-6300 Zug, Tel. 042-440550/660, Telefax 042-415770, Telex: 862329 mut ch*
***USA:*** *M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (415) 366-3600, Telex 752-351*
***Österreich:*** *Markt&Technik Ges. mbH, Große Neugasse 28, A 1040-Wien Telefon: 0222/5871393, Telex: 047-132532*
***Anzeigen-Auslandsvertretung:***
***England:*** *F. A. Smyth&Associates Limited, 23a, Aylmer Parade, London, N2 OPQ. Telefon:0044/1/3405058, Telefax: 0044/1/3419602*
***Israel:*** *Baruch Schaefer, Haeskel-Str. 12, 58348 Holon, Israel, Tel. 00972-3-5562256*
***Taiwan:*** *Aim International Inc., 4F.-1, No. 200, Sec. 2, Hsin-I Rd.; Taipei, Taiwan, R.O.C., Tel. 00886-2-7548631, -7548633, Fax 00886-2-7548710*
***Korea:*** *Young Media Inc., C.P.O. Box: 6113, Seoul/Korea, Tel. 0082-2-7564819, /-7742759, Fax 0082-7575789*
***USA:*** *M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (415) 366-3600, Telex 752-351*

***Vertriebsdirektor:*** *Uwe W. Hagen*

***Vertriebsmarketing:*** *Petra Schlichthärle (703)*

***Vertrieb Handel:*** *Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: ip Internationale Presse, Hauptstätter Straße 96, 7000 Stuttgart 1, Tel. 0711/6483-110*

***Bezugsmöglichkeiten:*** *Leser-Service: Telefon (089) 4613-366. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.*

***Verkaufspreis:*** *Das Einzelheft kostet DM 16,–*

***Produktion:*** *Technik: Klaus Buck (Ltg./180), Wolfgang Meye[illegible]tellv./887); Herstellung: Otto Albrecht (Ltg./917)*

***Druck:*** *SOV Graphische Betriebe, Laubanger 23, 8600 Bamb[illegible]*





***Sonderdruck-Dienst:*** *Alle in dieser Ausgabe erschienenen [illegible]nd in Form von Sonderdrucken zu erhalten. Anfragen an Reinhard Jarczok, Tel. 0[illegible]5, Fax 4613-774.*





***Vorstand:*** *Otmar Weber (Vors.), Bernd Balzer, Richard Kerler*

***Verlagsleitung:*** *Wolfram Höfler*

***Direktor Zeitschriften:*** *Michael M. Pauly*

***Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:*** *Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 522052, Telefax 089/4613-100*

***ISSN 0931-8933***

***Telefon-Durchwahl im Verlag:***
*Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/4613 und dann die Nummer, die in den Klammern hinter dem jeweiligen Namen angegeben ist.*

etc. Der Paramter P hinter »N« dient wie beim Find-Befehl zur Angabe der Page. Siehe hierzu auch den P-Befehl.

Achtung: Die angesprungenen Zeilennummern bei den Befehlen .ON und .GOTO werden nicht berücksichtigt und müssen von Hand angepaßt werden! Eine solche Einschränkung wurde in Kauf genommen, damit das gesamte Programm nicht zu lang wird, um es in einem 2764-EPROM unterzubringen.

**O (Old)** <F6>
holt nach einem Reset oder C-Befehl den alten Quelltext zurück. Dabei wird automatisch der B-Befehl zur Anzeige der Speicherkonfiguration aufgerufen. Der O-Befehl läßt sich auch durch <F6> aufrufen.

**P (Page)**
setzt einen Arbeitsbereich (Page, siehe Find- und Number-Befehl). Syntax:

P N, Zeile 1, Zeile 2

Der Parameter »N« (Nummer der Page) liegt zwischen 0 und 15. »Zeile 1« gibt die erste Zeile der Page an und »Zeile 2« die letzte. Zum Beispiel setzt P1, 0, 99 die Page 1 auf den Zeilenbereich 0 bis 99. Direkt nach Initialisierung des Assemblers ist die Page 0 auf den Zeilenbereich 0 bis 65535 gesetzt, sie umfaßt also den gesamten Quelltext. Man sollte Page 0 daher zweckmäßigerweise nicht umdefinieren.

Eine Page wird auf den Bildschirm ausgegeben, indem man die <←>-Taste drückt, darauf die Page-Nummer eingibt und dann <RETURN> drückt. Auf einzelne Pages greifen die Befehle »F, N und R« zu. Gibt man dort statt einer 0 eine andere Nummer von 1 bis 15 ein, so bezieht sich der Befehl nicht auf den gesamten Quelltext, sondern eben nur auf die angegebene Page.

**Q (Ausgabe der Page-Definitionen)**
gibt die Definitionen aller 16 Pages auf den Bildschirm aus. Hieraus kann man ersehen, welche Pages bereits definiert wurden, um nicht ungewollt eine Page umzudefinieren.

**R (Replace)**
dient zum Suchen und Ersetzen von Zeichenfolgen im Quelltext. Syntax:

R P, "Ersatzwort", "Suchwort"

Der R-Befehl ersetzt »Suchwort« durch »Ersatzwort« in der angegebenen Page (anstelle der 0 kann auch eine andere Nummer von 1 bis 15 eingegeben werden). In »Suchwort« ist das Fragezeichen als Joker erlaubt. Da »Ersatzwort« auch leer sein darf, lassen sich mit diesem Befehl auch Zeichenketten löschen. Im übrigen gelten beim R-Befehl dieselben Einschränkungen wie beim F-Befehl.

**S (Save)**
dient zum Speichern von Quelltext auf Diskette. Syntax:

S "Filename"

Der gesamte Quelltext wird in ein PRG-File auf Diskette geschrieben. Anmerkung: Falls man zuvor den Basic-Befehl LOAD zum Laden eines Monitors verwendet hat und dadurch der Quelltextende-Zeiger verbogen wurde, sollte man vor Eingeben des S-Befehls den O-Befehl aufrufen. Dieser setzt den Quelltext-Ende-Zeiger wieder richtig.

**T (Tabulatoreinstellung)**
setzt die Tabulatoren 0 beziehungsweise 1 (X) auf neue Werte. Syntax:

T X, Wert

Die Tabulatoren nehmen Einfluß auf das Format, in dem Quelltextzeilen gelistet werden. Tabulator 0 bestimmt, wieviele Spalten für Labels belegt werden, und Tabulator 1 gibt an, ab welcher Spalte die Kommentare am Ende einer Zeile gelistet werden. Als erste Position innerhalb einer Quelltextzeile gilt dabei die Spalte unmittelbar hinter der Zeilennummer. Voreingestellt sind: T0,10 und T1,24.

**V (Verify)**
entspricht dem Basic-Befehl VERIFY. Syntax:

V "Filename"

**X (Assemblieren)** <F3>
startet die Assemblierung. Anstelle von »X« ist auch der Basic-Befehl RUN verwendbar. Dieser Befehl wird auch durch <F3> abgerufen.

**Y (Ausgabe der Symboltabelle )**
gibt nach beendeter Assemblierung die Symboltabelle auf dem Bildschirm aus. Eine Ausgabe auf den Drucker wird erreicht durch:

OPEN1,4,7: CMD1 und anschließenden Y-Befehl.

**@ (Disk-Status und Floppy-Befehle)**
zeigt den Status des Diskettenlaufwerks an und leitet Diskettenbefehle (I, R, N, V) ein.

## Speeddos-kompatible Funktionstastenbelegung

Auf den Funktionstasten <F1>, <F3>, <F5>, <F7> liegen die Editor-Befehle »E, X, L und I«. Man beachte die Übereinstimmung zu der Speeddos-Belegung LIST, RUN, LOAD und Directory. Auf den Funktionstasten <F2>, <F4>, <F6> und <F8> liegen die Editor-Befehle »B (Speicheranzeige), Y (Ausgabe der Symboltabelle), O (Old) und @(Disk-Status)«.

Die Funktionstastenabfrage wurde in den Interrupt eingebunden. Das hat zur Folge, daß Programme, die mit Interrupts arbeiten, diese Abfrage ausschalten. Man muß nur dafür sorgen, daß nach Beendigung eines solchen Programms der Interrupt-Vektor in $0314/$0315 wieder auf den Interrupt zur Funktionstastenabfrage gesetzt wird. Dies kann auch mit den Tasten <RUN/STOP RESTORE> geschehen. Die eleganteste Lösung dafür ist, zu Beginn des Programms den alten Interruptvektor zu retten und am Ende wiederherzustellen. Es geht aber auch so, daß man am Ende der Interrupt-Benutzung den Vektor statt auf $EA31 direkt auf die entsprechende Adresse setzt, welche der Einsprungübersicht (Tabelle 4) zu entnehmen ist.

Nun soll an einem konkreten Beispiel gezeigt werden, wie man einen Quelltext eingibt.

Geben Sie A 100, 10 und dann das folgende Programm ein (ohne die Zeilennummern, diese werden durch den Auto-Befehl ja vorgegeben). Achten Sie dabei darauf, daß <SPACE> bedeutet: Einmal die SPACE-Taste drücken.

```
100.BASE $6000
110.START $6000
120.MACRO PRINT <SHIFT SPACE> TEXT
130 <SPACE> LDA #< (TEXT)
140 <SPACE> LDY #> (TEXT)
150 <SPACE> JSR $AB1E
160.ENDMACRO
170 <SPACE> PRINT <SPACE> MESSAGE
180 <SPACE> RTS
190MESSAGE <SPACE> .TEXT"<beliebige Mitteilung>"
```

Nun erscheint noch die Zeilennummer 200. Diese wird aber nicht mehr benötigt, wir drücken daher <SHIFT RETURN> und noch einmal <RETURN>. Mit <F1> kann man das Programm wieder ansehen. Es besteht im wesentlichen aus dem Makro »PRINT«, das eine beliebige Mitteilung auf dem Bildschirm ausgibt. Der verwendete Parameter »TEXT« ist eine Adresse, die auf den Textanfang

verweist, in unserem Fall symbolisch angegeben durch das Label »MESSAGE«. Der Aufruf für dieses Makro steht in Zeile 170.

Der Text in Zeile 190 wird zwischen Anführungszeichen genauso eingegeben, wie man das vom Basic-Befehl PRINT her gewohnt ist (inklusive aller Cursor-Steuerzeichen). Die eigentliche Text-Ausgabe erfolgt durch einen Einsprung in die entsprechende Routine des Betriebssystems ($AB1E).

Nun startet man den Assembler entweder durch <F3>, <X> <RETURN> oder auch durch »RUN« und <RETURN>. Sobald die Assemblierung beendet ist, erscheint die Meldung:

»<SPACE> FOR .START OR <RUN/STOP>«

Folgen Sie dieser Aufforderung und drücken <SPACE>, so wird der Bildschirm gelöscht und dann der Text aus Zeile 190 ausgegeben.

Es können also Speicheradressen wie gewohnt hexadezimal mit einleitendem $-Zeichen eingegeben werden. Im Beispiel wurde die Basis-Adresse auf $6000 gesetzt. Man hätte aber auch genausogut .BASE 24576 (also dezimal) eingeben können. Dezimalzahlen werden also nicht besonders gekennzeichnet. Ferner erkennt der Assembler auch Binärzahlen, die durch »%« eingeleitet werden. So ist %1010 = $A = 10.

Sehr wichtig beim Eingeben von Quelltexten ist die Unterscheidung von <SPACE> und <SHIFT SPACE>. Im Kapitel über Makro-Programmierung wird darauf noch eingegangen. Hier noch ein wichtiger Hinweis: Es kann vorkommen, daß beim Assemblieren eine auf den ersten Blick syntaktisch korrekte Zeile mit einem Fehler quittiert wird. Schuld daran ist in vielen Fällen ein verstecktes <SHIFT SPACE>, das leider beim Original-Zeichensatz von einem <SPACE> nicht zu unterscheiden ist. Eine Hilfe für das Programmieren mit Giga-Ass wäre daher ein 2732-EPROM mit einem modfizierten Zeichensatz, bei dem <SHIFT SPACE> als kleiner Punkt (wie bei Vizawrite) auf dem Bildschirm erscheint. Dieses EPROM müssen Sie anstelle des originalen Character-ROM einsetzen.

### Die Token

Jede eingegebene Zeile wird nach <RETURN> gepackt und erst dann im Speicher abgelegt. Dabei werden die Assembler-Befehle wie LDA, STA etc. (die sogenannten Mnemonics) in Token umgerechnet. Statt nun tatsächlich »LDA«, »STA« einzugeben, kann man auch gleich das Grafiksymbol eintippen, welches den gleichen Zeichencode wie das entsprechende Token besitzt. Zum Beispiel kann man statt »LDA« direkt <SHIFT J> eingeben und statt STA die Tasten <SHIFT R> drücken. In Tabelle 3 sind die Token für alle Mnemonics verzeichnet.

### Die Pseudo-Befehle

Die Pseudo-Befehle sind Anweisungen an den Assembler, die den Verlauf der Assemblierung steuern. Pseudo-Befehle beginnen immer mit einem Punkt. In Giga-Ass werden auch die Pseudo-Befehle als Token gespeichert. Das hat zur Folge, daß – wie bei den Assembler-Befehlen – eine abgekürzte Eingabe möglich ist. Doch aufgrund der Tatsache, daß die Grafiksymbole für je zwei unterschiedliche Codes stehen, treten Überschneidungen mit den Abkürzungen für die Assembler-Befehle auf. Dieses Problem wurde folgendermaßen gelöst: An jeder Stelle, wo ein Befehl wie LDA eingegeben wird, kann man auch das entsprechende Token direkt als Grafiksymbol eingeben.

Für die Pseudo-Befehle wurde nun eine andere Form der Abkürzung gewählt. Vermutlich ist jedem die Methode bekannt, Basic-Befehlsworte abgekürzt einzugeben: LIST kürzt man ab durch <L> <SHIFT I> und LOAD durch <L> <SHIFT O>. Es gibt aber auch Basic-Befehlsworte,

| Adresse | Inhalt |
|---|---|
| $8000 | RESET-Vektor, zeigt auf Kaltstart $849D |
| $8002 | NMI-Vektor, zeigt auf NMI-Routine $9FAE |
| $8004 | ROM-Kennung »CBM80« |
| $8009 | INIT-Vektor, durch JMP ($8009) Initialisierung |
| $800B | Tabelle der Mnemonic-Texte (CPX,...,BEQ) |
| $80B3 | Tabelle der zu den Mnemonics gehörigen Codes |
| $80EB | diverse Tabellen zur Mnemonic - Code-Wandlung |
| $8110 | Tabelle der binären Operatoren »+, −, *, /, ↑, A, O, >, =, < |
| $811A | Texte der Assembler-Fehlermeldungen |
| $81E5 | Adreßtabelle der einzelnen Fehlermeldungs-Texte |
| $8207 | Texte der Pseudo-Befehle CALL bis NOEXP |
| $827A | Vektortabelle für Einsprünge in Pseudo-Routinen |
| $82AC | Tabelle der Editorkommandos M, V,...,X, Y |
| $82C1 | Vektortabelle für Einsprünge in Editor-Routinen |
| $82EB | Einschaltmeldung GIGA-ASS |
| $832C | Texte zur Benutzerführung |
| $848C | Prompt »GIGA-ASS READY« |
| $849D | Kaltstart, wird bei RESET aufgerufen |
| $84AA | Initialisierung des Assemblers, beginnt mit SEI |
| $84AB | ROM-Bereich $8000 in RAM $8000 umkopieren |
| $84C4 | Interrupt-Vektor auf Funktionstastenabfrage setzen |
| $84CF | Basic-Ende auf $8000 setzen |
| $84D9 | Basic initialisieren |
| $84DF | Basic-Vektoren ab $0300 umlenken |
| $8507 | Tabulator-Initialisierung T0, 10 und T1, 24 |
| $8511 | Page 0 auf Zeile 0 bis 65535 setzen |
| $8519 | Einschaltmeldung ausgeben |
| $8537 | Ausgabe des Prompts |
| $8544 | Warmstart-Einsprung |
| $854A | Abschluß einer Merge-Operation |
| $8557 | Fehlermeldung ausgeben |
| $85C1 | Auswerten von Ausdrücken |
| $85F4 | Erkennen der Einleitsymbole »%, $, −, !, +, >, < |
| $8623 | Einlesen von Binärzahlen |
| $8647 | Erkennen von !N! |
| $8658 | Fehlermeldung beim Auswerten von Ausdrücken |
| $865D | Erkennen der binären Operatoren von $8110 |
| $868E | Low-/High-Byte Bildung |
| $86A6 | Bestimmen der Adressierungsart eines Befehls |
| $8710 | Erkennen auf indizierte Befehle |
| $8781 | Berechnen des Maschinencodes für ein Mnemonic |
| $882E | Auswerten von Branch-Befehlen (bei relativen Sprüngen) |
| $8870 | Einlesen von Hexadezimalzahlen |
| $88B2 | Beginn der Symboltabellen-Verwaltung |
| $88BA | Holen eines Symbols |
| $88C8 | Illegales Symbol erkannt |
| $88DB | Erkennen des Trennzeichens <←> |
| $88E6 | Sucht, ob Symbol schon in Tabelle eingetragen ist |
| $8955 | Symbol nicht gefunden - sieht auf dem Stack nach |
| $897D | Rücksprungadresse auf Stack ergab »Makro undefiniert« |
| $8985 | Bei der Auswertung von Ausdrücken Symbol undefiniert |
| $89A4 | Ansonsten wird neues Symbol eingetragen |
| $89E6 | Symbol gefunden |
| $8A0F | Wert des Symbols in Fließkomma-Akku schreiben |
| $8A19 | Symboltabelle voll! |
| $8A25 | Holt neue Quelltextzeile |
| $8A4C | Ende eines Pass |
| $8BC6 | Ordnungszahl zurücksetzen |
| $8BD2 | Direkt-Befehl auswerten |
| $8BFE | RUN beziehungsweise X: Start der Assemblierung |
| $8C24 | Startet Uhr (Assemblierungsdauer) |
| $8C72 | Holt Adressen aller Makro-Definition in Symboltabelle |
| $8CC0 | Nächsten Pass einleiten |
| $8CDC | Einsprungpunkt: nächste Quelltextzeile assemblieren |
| $8DDC | Pseudo-Befehl-Verteiler |
| $8E0A | .GLOBAL |
| $8E19 | .EQUATE |
| $8E6A | .BASE |
| $8E79 | .CODE |

**Tabelle 4. Einsprungadressen von Giga-Ass. Wollen Sie Giga-Ass so finden Sie hier wichtige Basisinformationen.**

| Adresse | Inhalt |
|---|---|
| $8E9B | .BYTE |
| $8EC1 | .TEXT |
| $8F2F | .WORD |
| $8F9F | .DS |
| $8FE1 | .OBJECT |
| $903F | .END |
| $9054 | .MACRO überspringt Makro-Definition |
| $907A | .CALL springt in Makro-Definition |
| $917B | .ON |
| $9194 | .GOTO |
| $919A | .IF |
| $91A7 | .ELSE |
| $91CD | .ENDIF |
| $91E5 | Legt Maschinencode für eine Quelltextzeile ab |
| $9217 | Erhöht Adreßpegel |
| $924C | Holt nur Zahl aus Quelltext |
| $925A | Holt Zahl oder Character aus Quelltext |
| $926A | Holt Character (String der Länge 1) aus Quelltext |
| $9281 | Erkennt auf Token |
| $92E0 | Gibt Uhrzeit aus (nach erfolgter Assemblierung) |
| $9332 | .LISTING |
| $936A | .PAGE |
| $9379 | Gibt Hexadezimalzahl aus |
| $9390 | Listet Maschinencodes |
| $93CD | Listet Quelltextzeile, falls .LISTING aktiv |
| $93EB | Prüft auf Seitenwechsel bei .PAGE |
| $9422 | .SYMBOLS |
| $9442 | Am Ende der Assemblierung Symboltabelle ausgeben |
| $946C | .STOP |
| $947C | Blank-Ausgabe |
| $949F | .NOCODE |
| $94A6 | .START |
| $94B2 | .NOEXP |
| $94BF | Warmstart des Editors, holt Editor-Befehl |
| $953E | Listet Quelltextzeile nach Eingabe |
| $956D | Fügt Pseudo-Token-Code ein |
| $95C0 | Packt und wandelt eingegebene Zeile |
| $96E8 | Gibt Zeilennummer im Auto-Modus vor |
| $9749 | OFF: schaltet Auto-Modus aus |
| $974F | A-Befehl |
| $9773 | Holt eingegebenen Zeilen-Bereich |
| $97D8 | D-Befehl |
| $9874 | E-Befehl |
| $988E | P-Befehl setzt Page-Parameter |
| $98BD | Holt Page-Parameter |
| $98E2 | Gibt Page auf Bildschirm aus |
| $98F3 | N-Befehl |
| $995B | M-Befehl |
| $997D | Holt Filenamen |
| $9990 | V-Befel |
| $9993 | L-Befehl |
| $99A0 | S-Befehl |
| $99A9 | F-Befehl |
| $9A18 | Holt Zeilennummer aus Quelltext |
| $9A23 | Gibt Zeilennummer rechtsbündig aus |
| $9A44 | Gibt eine Quelltextzeile formatiert aus |
| $9B13 | SPACE hält Listig an |
| $9B33 | R-Befehl |
| $9C4A | I-Befehl |
| $9CBE | @-Befehl |
| $9D0A | Y-Befehl |
| $9E13 | O-Befehl |
| $9E2D | B-Befehl |
| $9EED | G-Befehl |
| $9F08 | T-Befehl |
| $9F24 | Q-Befehl |
| $9F71 | IRQ-Routine: hierauf muß der IRQ-Vektor zeigen |
| $9FA6 | Funktionstastenbelegung »E, X, L, I, B, Y, 0, @« |
| $9FAE | NMI-Routine |
| $9FFF | Letztes Byte. Der Code belegt die vollen 8 KByte! |

**für spezielle Zwecke umschreiben oder erweitern,**

die in den ersten beiden Buchstaben gleich sind, beispielsweise READ und RESTORE. Hier muß dann <R>, <E>, <SHIFT A> beziehungsweise <R>, <E>, <SHIFT S> eingegeben werden.

Das gleiche Prinzip wurde für die Pseudo-Befehle von Giga-Ass eingesetzt: .BASE kürzt man ab durch <.> <B> <SHIFT A>, .GOTO durch <.><G> <SHIFT O>. Da sich aber die meisten Pseudo-Befehle bereits im ersten Buchstaben unterscheiden, ist es im Gegensatz zu Basic zulässig, bereits den ersten Buchstaben geSHIFTet einzugeben. So wird .MACRO abgekürzt, indem man <.><SHIFT M> eingibt.

## 25 Pseudo-Befehle

Bei der folgenden Aufstellung aller 25 Pseudo-Befehle wird angegeben, welches die kürzeste Abkürzung für den entsprechenden Befehl ist.

**.CALL**
dient zum Aufruf von Makros. Dieser Pseudo-Befehl nimmt eine besondere Stellung ein, da er in dieser Form gar nicht im Quelltext erscheint. Näheres dazu weiter unten.

**.MACRO** <.><SHIFT M>
leitet eine Makro-Definition ein.

**.ENDMACRO** <.><SHIFT E>
beendet eine Makro-Definition.

**.GLOBAL** <.><SHIFT G>
definiert ein globales Symbol (Label). Syntax:
.GLOBAL Symbolname = Wert
Zum Beispiel wird mit:
.GLOBAL CHROUT=$ffd2
die Kernel-Routine zur Ausgabe einzelner Zeichen mit dem symbolischen Namen CHROUT benannt. Ein Label muß dann als global definiert werden, wenn es sowohl in einem Makro als auch im Hauptprogramm verwendet werden soll.

**.EQUATE** <.><E> <SHIFT Q>
definiert ein lokales Symbol. Syntax:
.EQUATE Symbolname = Wert

Lokale Symbole sind für Schleifen und Sprungbefehle zu verwenden. Außerdem werden sie bei jedem Aufruf eines Makros mit Parametern implizit angelegt.

**.BYTE** <.><SHIFT B>
fügt einzelne Bytewerte (Werte von $00 bis $FF) in den Quelltext ein. Mehrere Bytewerte werden duch Kommata voneinander getrennt. Die Werte sind wahlweise als dezimale, hexadezimale oder binäre Zahlen, als Symbole oder als Strings der Länge 1 einzugeben. Zum Beispiel werden durch:

```
.byte 32,$20,%100000," "
```

vier Leerzeichen mit dem ASCII-Code 32 erzeugt.

**.WORD** <.><SHIFT W>
fügt Adressen (16-Bit-Werte im Bereich von $0000 bis $FFFF) in den Quelltext ein. Mehrere Adressen sind durch Kommata voneinander zu trennen. Die Adressen werden in der standardmäßigen Folge Low-, High-Byte in den Objektcode aufgenommen.

**.DS** <.><SHIFT D>
DS steht für »Define Storage« und dient zur Reservierung von Speicherbereichen. Syntax:
.DS »n«

Die Anzahl »n« der zu reservierenden Bytes kann zwischen 0 und 255 liegen. Vor .DS kann natürlich ein Label stehen, um den so erzeugten Speicherbereich anzuspre-

chen. Im Gegensatz zum DS-Befehl von anderen Assemblern, die einfach den Adreßpegel erhöhen und den Speicher undefiniert lassen, initialisiert Giga-Ass den Speicherbereich mit Null. Die so erzeugten Null-Bytes werden auch ins Übersetzungs-Protokoll mit aufgenommen.

**.TEXT** <.> <SHIFT T>
erlaubt das Einfügen von Texten in den Quelltext. Die einzelnen Zeichen des Textes werden als ASCII-Codes im Speicher abgelegt. Es gibt zwei Syntax-Varianten:

a) .TEXT "Zeichenfolge"

Mit abschließendem Anführungszeichen fügt man hinter das letzte ASCII-Zeichen noch ein zusätzliches Nullbyte als Begrenzer. Das hat den Vorteil, daß so definierte Texte ohne weiteres mit der PRINT-Routine $AB1E ausgegeben werden können.

b) .TEXT "Zeichenfolge

Ohne abschließendes Anführungszeichen legt man nur die Zeichenfolge – ohne Nullbyte – im Objektcode ab. Nachteil ist, daß in dieser Variante abschließende Leerzeichen nicht eingegeben werden können. In diesem Falle, muß man sich mit eventuell mehreren ».BYTE 32«-Befehlen behelfen.

Die Taste <←> wurde ausgewählt, um einen Carriage-Return-Code ($0D) zu erzeugen. Damit ist es möglich, innerhalb eines .TEXT-Befehls in die nächste Zeile zu springen. Sonst müßte man das mit .BYTE $0D-Befehlen eingeben. Ein Nachteil dabei ist, daß nun das Zeichen <←> innerhalb von .TEXT-Befehlen nicht mehr erzeugt werden kann. Es muß als ».BYTE $5F« eingegeben werden.

**.OBJECT** <.> <SHIFT O>
lenkt die Objektcode-Ausgabe auf Diskette um. Syntax:
.OBJECT "filename,p,w"

Mit diesem Befehl ist es möglich, direkt auf Diskette zu assemblieren. Der .OBJECT-Befehl öffnet das File mit dem Namen »filename« zum Schreiben. Sollte dieses bereits vorhanden sein, so gibt Giga-Ass die Meldung FILE EXISTS ERROR aus. War das Öffnen des Files erfolgreich, so werden als erste Bytes die Basis-Adresse als Startadresse für den Objektcode im Low-, High-Byte-Format in das File geschrieben. So wird erreicht, daß das Objektprogramm an die richtige Stelle im Speicher geladen wird.

Achtung: Wird dieser Befehl verwendet, so muß der Quelltext unbedingt als letzten Befehl einen .END-Befehl enthalten! Dieser schließt das File. Sollte man diesen Befehl vergessen haben, so ist nach Beendigung der Assemblierung vor dem nächsten Diskettenzugriff das Objekt-File mit CLOSE 14 von Hand zu schließen.

Achtung: Die Befehle .OBJECT und .START (siehe unten) schließen einander aus. Der Maschinencode steht durch .OBJECT nicht im Speicher und kann daher auch nicht direkt nach der Assemblierung gestartet werden.

**.BASE** <.><B><SHIFT A>
setzt die Basis-Adresse für den Objektcode. Syntax:
.BASE Adresse

Dieser Befehl sollte zu Beginn des Quelltextes stehen.

**.CODE** <.><C><SHIFT O>
bestimmt, in welchen Speicherbereich der Objektcode abgelegt wird. Syntax:
.CODE Adresse

Der Objektcode wird ab der angegebenen Adresse in den Speicher geschrieben. Dieser Befehl ist nur notwendig, wenn der Objektcode an einer anderen Stelle des Speichers abgelegt werden soll als derjenigen, wo er später als lauffähiges Programm liegen soll (beispielsweise wenn der gewünschte Speicherbereich derzeit von einem anderen Programm belegt ist.) Ist im Quelltext kein .CODE-Befehl vorhanden, so wird der Objektcode ab der Basis-Adresse (.BASE-Befehl) im Speicher abgelegt.

**.ON**
ermöglicht einen bedingten Sprung. Syntax:
.ON Ausdruck, Zeilennummer

»Ausdruck« ist ein Vergleich, der –1 für wahr und 0 für falsch liefert. Ergibt »Ausdruck« als Resultat –1 (also wahr), so erfolgt der Sprung zu der angegebenen Zeilennummer; andernfalls wird mit der Assemblierung in der nächsten Zeile fortgefahren.

Als Zeilennummer wird auch ein zu berechnender oder durch .EQUATE festgelegter Ausdruck integriert. Mit einer Kombination von .EQUATE- und .ON-Befehl lassen sich sehr leicht Assemblerschleifen realisieren. Zwei Beispiele:

a) Es soll eine Tabelle erzeugt werden, in der alle Zahlen von 0 bis 99 in ihrer numerischen Reihenfolge als Bytes eingetragen sind. Wir erreichen dies durch folgendes Programm:

```
10.BASE $6000
20.EQUATE X=0
30.BYTE X
40.EQUATE X=X+1
50.ON X <100, 30
```

b) Wir möchten alle Buchstaben von »A« bis »Z« auf den Bildschirm ausgeben, aber kein Indexregister verwenden. Als Assemblerschleife läßt sich das so programmieren:

```
10.BASE $6000
20.START $6000
30.EQUATE C=65
40 <SPACE> LDA #C
50 <SPACE> JSR $FFD2
60.EQUATE C=C+1
70.ON C< 91,40
80 <SPACE> RTS
```

**.GOTO** <.><G> <SHIFT O>
erzeugt einen unbedingten Sprung. Syntax:
.GOTO Ausdruck

Der »Ausdruck« wird als Zeilennummer ausgewertet. Anschließend wird die Assemblierung bei dieser Zeilennummer fortgesetzt. In Verbindung mit dem .STOP-Befehl läßt sich ein Quelltext in einzelnen Abschnitten assemblieren, um einen Fehler einzugrenzen.

**.IF** <.> <SHIFT I>
.ELSE <.> <E> <SHIFT L>
**.ENDIF** <.> <END> <SHIFT I>
ermöglichen eine bedingte Assemblierung. Syntax:
.IF Ausdruck

Wird der »Ausdruck« hinter .IF als 0 – also falsch – ausgewertet, wird die Assemblierung hinter .ELSE fortgesetzt. Wird kein .ELSE gefunden, wird die Assemblierung hinter .ENDIF fortgesetzt.

Wurde der »Ausdruck« jedoch als –1 – also wahr – ausgewertet, so wird die Assemblierung fortgesetzt, bis auf .ELSE gestoßen wird. Darauf wird .ENDIF gesucht und dahinter die Assemblierung fortgesetzt.

**.SYMBOLS** <.><SHIFT S>
Syntax:
.SYMBOLS lfn, dev, sa, "filename"

Der .SYMBOLS-Befehl bewirkt, daß am Ende des Assemblierungsvorgangs die Symboltabelle ausgegeben wird. Die Parameter hinter .SYMBOLS entsprechen denen des normalen OPEN-Befehls in Basic. Die Symboltabelle ist fest auf 40 Zeichen pro Zeile formatiert und eignet sich daher auch für die Bildschirm-Ausgabe. Die Ausgabe auf dem Bildschirm wird durch Drücken der SPACE-Taste angehalten und wieder fortgesetzt oder durch <RUN/STOP> abgebrochen.

Beispiele:
.SYMBOLS 1,3,0 gibt die Symboltabelle auf den Bildschirm aus.
.SYMBOLS 1,4,0 gibt die Symboltabelle auf den Drucker aus.
.SYMBOLS 2,8,2,″table,s,w″ schreibt die Symboltabelle in ein sequentielles File mit dem Namen ″table″.

**.LISTING** <.> <SHIFT L>
Syntax:
.LISTING lfn, dev, sa, ″filename″

Der .LISTING-Befehl bewirkt, daß während der Assemblierung ein Übersetzungs-Protokoll auf das angegebene Gerät ausgegeben wird. Achtung: Sollten .LISTING und .SYMBOLS zusammen verwendet werden, so müssen sich die logischen Filenummern »lfn« unterscheiden, sonst gibt der Assembler die Meldung FILE OPEN ERROR aus. Die Parameter werden wie bei .SYMBOLS gesetzt.

**.END**
bewirkt ein Schließen des Objekt-Files (siehe .OBJECT). Etwaiger Quelltext nach .END wird bis zum nächsten .OBJECT in den Speicher assembliert.

**.STOP** <.> <S> <SHIFT T>
bewirkt einen Abbruch der Assemblierung - jedoch erst im dritten Pass. Dadurch ist man sicher, daß alle Labels und Makros zur Verfügung stehen. Der Objektcode wird jedoch nur bis .STOP erzeugt.

**.PAGE** <.> <SHIFT P>
Syntax:
PAGE n

Dieser Befehl bewirkt eine Seitenformatierung. Nach »n« Quelltextzeilen wird bei Bildschirmausgabe der Bildschirm gelöscht, beim Drucker ein Form Feed gesendet.

**.NOCODE** <.> <SHIFT N>
unterdrückt das Ablegen von Objektcode im Speicher. Sinnvoll ist dies beim direkten Assemblieren auf Diskette.

**.START** <.> <S> <T> <SHIFT A>
Syntax:
.START Adresse

Wenn man diesen Befehl eingegeben hat, so wird nach beendeter Assemblierung zur angegebenen Adresse gesprungen, nachdem auf die SPACE-Taste gewartet und der Bildschirm gelöscht wurde. Das aufgerufene Programm muß mit RTS enden, damit ein Rücksprung in den Assembler erfolgt. Diesen Befehl kann man in den Quellcode einbinden, um einen Probedurchlauf zu erzeugen.

**.NOEXP** <.> <N> <O> <SHIFT E>
unterdrückt die Makro-Expansion im Übersetzungs-Protokoll, was bedeutet, daß der innerhalb von Makroaufrufen erzeugte Objektcode nicht gelistet wird. Besonders bei mehrfach aufgerufenen Makros dient dies zur Übersichtlichkeit im Protokoll.

Die Token für die Pseudo-Befehle sind in Tabelle 2 angegeben. Angewendet werden diese Token bei den Editor-Befehlen F und R. Soll beispielsweise nach jedem Auftreten des Pseudo-Befehls .MACRO gesucht werden, so lautet der Befehl: F0,″<CBM K>″.

Für das Eingeben von Quelltextzeilen gilt folgende Sonderregelung:

In der ersten Spalte hinter der Zeilennummer dürfen die Pseudo-Befehle auch als Token eingegeben werden. Steht jedoch in der betreffenden Zeile ein Label oder rückt man den Pseudo-Befehl durch ein Leerzeichen ein, so werden nur noch Token für Assembler-Befehle akzeptiert.

Diese Sonderregelung ist notwendig wegen der Überlappung bei den Grafiksymbolen, die für je zwei unterschiedliche Codes stehen. Dabei müssen die Token für Pseudo-Befehle stets am Anfang einer Zeile stehen.

In der ersten Spalte hinter der Zeilennummer dürfen statt den normalen Token auch die sogenannten Zusatz-Token eingegeben werden. Bei diesen Token handelt es sich ausschließlich um geSHIFTete Tasten (Tabelle 2, Taste 2).

## Die Makro-Programmierung

Makros sind meist kürzere Befehlsfolgen, die im Quelltext häufiger vorkommen und deshalb zu einer Einheit zusammengefaßt werden. Zu jedem Makro gehört ein Name, mit dem es aufgerufen wird. An jedes Makro lassen sich beliebig viele Parameter übergeben, deren aktueller Wert dann bei der Assemblierung eingesetzt wird. Makro-Definitionen sind an beliebiger Stelle im Quelltext erlaubt. Alle Makro-Namen sind global in dem Sinne, daß innerhalb eines Makros jedes andere aufgerufen werden kann. Alle Parameter und Makro-internen Label sind lokal. Das heißt, verschiedene Makros dürfen durchaus Label beziehungsweise Parameter gleichen Namens verwenden.

Sehen wir uns die prinzipielle Form einer Makro-Definition anhand eines Beispiels an:

```
100.BASE $6000
110.START $6000
120.MACRO POKE <SHIFT SPACE> ADR, VAL
130.<SPACE> LDA #VAL
140 <SPACE> STA ADR
150.ENDMACRO
160 <SPACE> POKE <SPACE> 53280,0
170 <SPACE> RTS
```

Der .MACRO-Befehl wird gefolgt vom Makro-Namen und einer Parameterliste, wobei dazwischen ein »SHIFT SPACE« stehen muß. Man nennt diese Parameter auch »formale Parameter«. Mehrere solcher Parameter werden durch Kommata getrennt. Die Reihenfolge der formalen Parameter ist willkürlich und bleibt Ihnen überlassen, weil es sich um eine Definition handelt. Nur müssen Sie sich später bei der Verwendung des Makros auch daran halten. In unserem Beispiel ist der Makro-Name »POKE« und die Parameter sind »ADR VAL«.

Die Quelltextzeilen, die direkt auf den .MACRO-Befehl folgen, definieren die Befehlsfolge, die bei jedem Makro-Aufruf im Objekt-Programm abgelegt wird. Innerhalb dieser Befehlsfolge tritt jeder Parameter an beliebigen Stellen auf, und zwar beliebig oft, jedoch mindestens einmal (sonst wäre der Parameter ja sinnlos). Abgeschlossen wird die Befehlsfolge durch den Pseudo-Befehl .ENDMACRO.

Wir betrachten den Makro-Aufruf aus Zeile 160.

```
POKE <SPACE> 53280,0
```

Auf den Makro-Namen »POKE« folgt die Liste der sogenannten »aktuellen Parameter« 53280 und 0. Zwischen Makro-Name und den Parametern muß im Gegensatz zu Assembler-Befehlen und deren Operanden ein Leerzeichen stehen. Stößt der Assembler auf diesen Makro-Aufruf, so tut er folgendes: Er merkt sich die aktuellen Parameter 53280 und 0. Darauf springt er an die Stelle des Quelltextes, wo sich die Definition des POKE-Makros befindet. Nun setzt er die aktuellen Parameter 53280 und 0 für die formalen Parameter »ADR« und »VAL« ein: »ADR« wird auf 53280 gesetzt und »VAL« auf 0. Damit ergibt sich die Befehlsfolge: LDA #0, STA 53280, die bekanntlich die Rahmenfarbe auf Schwarz setzt. Diese Befehlsfolge wird im Objekt-Programm abgelegt. Danach stößt der Assembler auf .ENDMACRO und springt im Quelltext an die Stelle hinter dem Makro-Aufruf zurück.

Dazu ein Beispiel (das POKE-Makro sei nach wie vor definiert):

```
200.MACRO RAM
210 <SPACE> POKE 1,$36
220.ENDMACR
300.MACRO ROM
310 <SPACE> POKE 1,$37
320.ENDMACRO
```

Das RAM-Makro blendet das Basic-ROM aus; an dessen Stelle tritt der RAM-Bereich von $A000 bis $BFFF. Das ROM-Makro macht es wieder rückgängig.

Diese Makro-Definition nennt man »geschachtelt«, weil innerhalb eines Makros ein anderes Makro aufrufen wird. Die Schachtelungstiefe ist nur begrenzt durch die Größe des Prozessor-Stack.

**Zur Wahl der Makro-Namen**

Der Makroname sollte nicht zu lang gewählt werden, denn bei jedem Auruf muß man ihn so eintippen, wie man ihn definiert hat. Wichtiger ist jedoch, daß der Makro-Name

## Kurze Makronamen

nicht mit einem Mnemonic (also LDA, STA etc.) beginnt. Das ist zwar nicht verboten, hat aber trotzdem einen guten Grund. Ein Beispiel: Nehmen wir an, wir haben ein Makro »STATUS« definiert, das den Disk-Status ausgibt. Geben wir nun das Wort »STATUS« in eine Quelltextzeile ein, so vermag der Assembler prinzipiell nicht zu unterscheiden, ob es sich nun um ein Makro mit dem Namen »STATUS« handelt oder um den Assembler-Befehl STA mit »TUS« als symbolischem Operanden.

Sicher erkennen Sie das Problem: Soll die Tokenwandlung erfolgen oder nicht? Sie erfolgt, weil den Mnemonics immer der Vorrang eingeräumt wird. Trotzdem gibt es einen Weg, die Tokenwandlung zu unterdrücken: Man gibt vor dem Makro-Namen ein <SHIFT SPACE> ein. Dies ist das Token für den Pseudo-Befehl .CALL. Nur dann, wenn ein Makro-Name mit einem Mnemonic beginnt, muß es eingegeben werden, sonst ist es unnötig.

**Rechnungen im Quelltext**

Im Quelltext sind die vier Grundrechenarten Addition (+), Subtraktion (-), Multiplikation (*) und Division (/) sowie auch die Potenzierung (↑) erlaubt. Daneben gibt es noch die drei Vergleichsoperatoren »<, =, >« sowie vier logische Operatoren. Dies sind zunächst die beiden Operatoren »N« für das Einerkomplement (NOT) und »–« für das Zweierkomplement sowie zwei binäre Operatoren: »A« für das logische »AND« und »O«, das logische »OR«.

Für die Operatoren »N, A und O« gibt es eine Sonderregelung: Um diese Zeichen von Symbol-Zeichen zu trennen, müssen sie in Ausrufezeichen eingeschlossen werden. Bei NOT ist das obligatorisch, es muß immer als »!N!« geschrieben werden. Für AND und OR gilt: Nur wenn die Operanden Zahlen sind, kommt man ohne Ausrufungszeichen aus, beispielsweise ist:

1 A 1 = 1. Eine logische UND-Verknüpfung zwischen den beiden Symbolen »X« und »Y« erreicht man durch: X !A! Y.

Die Vergleichsoperatoren liefern als Wahrheitswerte -1 für wahr und 0 für falsch. Eine logische Negation (NOT) erreicht man durch den N-Operator. Es ist: !N! -1 = 0 und !N! 0 = -1. Damit lassen sich auch die Operatoren »< >«, »< =« und »= >« (ungleich, kleiner und größer gleich), die nicht direkt zur Verfügung stehen, realisieren: Es ist:

```
A<>B = !N! A = B
A <= B = !N! A > B
A >= B = !N! A < B
```

Beim Rechnen mit den Wahrheitswerten muß man darauf achten, daß man -1 und 0 zwar als Operanden für .IF und .ON verwenden, aber -1 nicht einem Symbol zugeweisen darf. Um Wahrheitswerte in Variablen zu speichern, muß man ins Zweierkomplement wandeln und anstelle mit -1 und 0 mit +1 und 0 rechnen. Zum Beispiel wird durch den Befehl .EQUATE C = -(A=B) das Symbol C auf 1 gesetzt, falls A=B gilt, und sonst auf 0.

**Das Adreßpegelsymbol »*«**

Das Symbol »*« liefert als Wert die Adresse zurück, an welcher der Assembler sich im Augenblick befindet. Diese Adresse nennt man auch den »Adreßpegel«. Sie gibt an, an welcher Adresse der Assembler das nächste Byte im Speicher ablegt.

Zur Verdeutlichung ein Beispiel. Geben Sie einmal folgendes Program ein:

```
10.BASE $6000
20.START $6000
30 <SPACE> LDX #0
40 MARKE TXA
50 <SPACE> STA $400,X
60 <SPACE> INX
70 <SPACE> BNE MARKE
80 <SPACE> RTS
```

Das Programm schreibt alle 256 Zeichen des Zeichensatzes ins obere Viertel des Bildschirms. Betrachten wir nun den Wert des Symbols »*« im Verlauf der Assemblierung:

```
30 *=$6000
40 *=$6002, MARKE = *
50 *=$6003
60 *=$6006
70 *=$6007
80 *=$6009
```

Man kann nun die hauptsächliche Verwendung des Symbols »*« erklären: Es dient zur Einsparung von Labeln bei relativen Sprüngen. Im Beispiel hat das Label »MARKE« den Wert $6002. Zugegriffen wird auf das Label in Zeile 70. Dort gilt * =$6007. Aber es ist auch *-5 = MARKE = $6002. Also hindert uns nichts daran, in Zeile 70 das Label »MARKE« durch den Ausdruck »*-5« zu ersetzen! Damit ist das Label »MARKE« in Zeile 40 überflüssig geworden; man kann es bedenkenlos wieder entfernen.

Vorteilhaft wird diese Variante in längeren Programmen, wo man es irgendwann leid ist, sich für jede Schleife einen neuen Label-Namen zu überlegen. Allerdings hat die Sache einen Haken: Man muß anfangen zu zählen, und zwar die Anzahl der Bytes, die durch den relativen Sprung übersprungen werden sollen. Hierfür gibt es folgende Regeln:

1. Bei Rückwärts-Sprüngen zählt man rückwärts die Anzahl der Bytes ab der Zeile direkt vor dem Sprungbefehl bis zur Zeile, in der das Sprungziel steht. In jeder Zeile zählt man dabei die Anzahl der Bytes, die der Befehl im Objekt-Programm belegen wird. Aufpassen müssen Sie vor allem bei der Unterscheidung von 2- und 3-Byte-Befehlen!

2. Bei Vorwärts-Sprüngen wird die Zeile, in der der Sprungbefehl steht, mitgezählt. Man zählt vorwärts bis zur Zeile unmittelar vor dem Sprungziel.

3. Besondere Vorsicht ist geboten, wenn man »*« als Parameter bei Makro-Aufrufen verwendet! Es wird dann der Adreßpegel im ersten Byte des expandierten Makros genommen. Hierzu ein Beispiel:

```
10.BASE $6000
20.START $6000
```

```
30.MACRO EQB <SHIFT SPACE> A,B
40 <SPACE> LDA A
50 <SPACE>  BEQ B
60.ENDMACRO
70 <SPACE> LDA # 0
80 <SPACE> STA 198
90 <SPACE> EQB <SPACE> 198,*
99 <SPACE> RTS
```

Dieses Programm wartet darauf, daß die Anzahl der gedrückten Tasten ungleich 0 wird. Der EQB-Makro-Aufruf entspricht der Befehlsfolge:

LDA 198, BEQ *-2.

**Das »&«-Symbol**

Schließlich kommen wir noch zum Symbol »&«. Es gibt an, daß die darauf folgende Zero-Page-Adresse indirekt angesprochen wird. Damit wird die normale Notation, bei der die Zero-Page-Adresse eingeklammert wird, abgekürzt.

So kann man z.B. statt LDA (ADR,X) schreiben LDA &ADR,X und STA (ADR),Y abkürzen durch STA &ADR,Y.

**Fehlermeldungen von Giga-Ass**

Tritt während der Assemblierung ein Fehler auf, so wird die Assemblierung abgebrochen und eine Fehlermeldung sowie die fehlerhafte Zeile ausgegeben. Zur Fehlerbehebung ist die Symbol-Tabelle von Nutzen. Ferner lassen sich mit dem Befehl »/«, welcher den PRINT-Befehl aus dem Basic ersetzt, einzelne Symbol-Werte auf dem Bildschirm ausgeben. Wurde der Fehler innerhalb eines Makros entdeckt, so stehen nur die Symbole innerhalb des Makros zur Verfügung. Geben Sie bitte

```
POKE $8B, 0: POKE $8C, 0
```

ein, um auf die globalen Symbole zuzugreifen.

## Eigene Fehlermeldungen

**SYNTAX ERROR**

Eine Quelltextzeile wurde nicht richtig gebildet. Beachten Sie bitte die richtige Reihenfolge Label, Assembler-Befehl, Operanden, Kommentar.

**ILLEGAL QUANTITY ERROR**

Ein Wert liegt außerhalb des zulässigen Bereichs. Byte-Werte liegen im Bereich $00 bis $FF, Wort- und Symbol-Werte im Bereich $0000 bis $FFFF. Alles, was darüber hinausgeht, führt zu dieser Fehlermeldung.

**TERM EVALUATION ERROR**

Ein Ausdruck konnte nicht berechnet werden. Beachten Sie hierzu das Kapitel über Berechnungen im Quelltext.

**TOKEN ERROR**

Zulässige Token liegen im Bereich $A0 bis $B8 für die Pseudos und $C0 bis $F7 für die Mnemonics. Treten andere Byte-Werte größer oder gleich $80 im Quelltext auf, so tritt dieser Fehler auf. Ausnahme: In der ersten Zeile steht ein SYS-Befehl mit dem Token $9E. In diesem Fall wird der Programmtext nicht assembliert, sondern als Maschinenprogramm betrachtet und mit dem SYS-Befehl gestartet. Damit lassen sich auch innerhalb des Assemblers Maschinenprogramme einladen und starten.

**INDEXING ERROR**

Ein Assembler-Befehl existiert nicht in der Kombination von gewählter Adressierungsart und gewähltem Register.

**LINE FORMAT ERROR**

Das Zeilenformat für Quelltextzeilen wurde nicht eingehalten. Beachten Sie, daß Kommentare am Schluß der Zeile mit Semikolon abgetrennt werden müssen.

**ADRESSING ERROR**

Ein Assembler-Befehl existiert nicht in der gewählten Adressierungsart.

**BRANCH ERROR**

Ein relativer Sprung führt über eine zu große Distanz (größer als 128 Byte). Ersetzen Sie diesen Branch-Befehl durch einen »Long Branch« mit Hilfe eines JMP-Befehls.

**UNDEFINED SYMBOL ERROR**

Ein Symbol wurde nicht vor seiner erstmaligen Verwendung definiert. Undefinierte Symbole haben den Wert $FFFF; dieser Wert darf also nie einem Symbol zugewiesen werden. Der Fehler tritt auch dann auf, wenn innerhab eines Makros ein lokales Label außerhalb der Makro-Definition angesprochen wird. Definieren Sie hierzu dieses Label als global vor der Definition des Makros.

**ILLEGAL SYMBOL ERROR**

Dieser Fehler tritt dann auf, wenn ein Symbol nicht mit einem Buchstaben beginnt. Beachten Sie, daß Groß-/Kleinschreibung bei Symbolen nicht erlaubt ist. Als Trennzeichen innerhalb von Symbolen kann <←> verwendet werden.

**SYMBOL TABLE FULL ERROR**

Die Symboltabelle ist mit 1170 Symbolen gefüllt, wenn dieser Fehler auftritt. Das passiert nur dann, wenn sehr viele Makros mit sehr vielen Parametern sehr oft aufgerufen werden – also praktisch nie.

**NO MACRO TO CLOSE ERROR**

Es kommt ein .ENDMACRO zuviel im Quelltext vor! Dieser Fehler tritt auch dann auf, falls vor einem .MACRO-Befehl ein Label steht. Dies ist so nicht erlaubt. Schreiben Sie das Label allein in eine freie Zeile direkt vor dem .MACRO-Befehl!

**DOUBLE LABEL ERROR**

Sie haben versucht, einen Label-Namen zweimal zu definieren. Geben Sie das zweite Mal einen anderen Namen ein. Kritisch ist es, wenn Sie innerhalb einer Assemblerschleife Label definieren. Dort ist es dann notwendig, mit dem »*«-Symbol zu arbeiten.

**PARAMETER ERROR**

Die Anzahl der Parameter in einem Makro-Aufruf stimmt nicht mit der Definition überein.

**RETURN ERROR**

Diese Fehlermeldung wird nur dann erzeugt, wenn Sie mit einem Assembler-Sprung mitten in eine Makro-Definition hineinspringen. Kontrollieren Sie die Zeilenummern in Ihren Sprungbefehlen.

**UNDEFINED MACRO ERROR**

Bei einem Makro-Aufruf wurde die zugehörige Makro-Definition nicht gefunden.

**MACRO NOT CLOSED ERROR**

zeigt einen grundlegenden Fehler an. Sie haben vergessen, Ihre Makro-Definition mit .ENDMACRO abzuschließen. Dieser Fehler tritt auch dann auf, wenn vor einem .ENDMACRO-Befehl ein Label steht. Dies ist nicht erlaubt. Schreiben Sie das Label allein in eine freie Zeile direkt vor dem .ENDMACRO-Befehl!

**IF-ELSE-ENDIF ERROR**

Achten Sie darauf, daß die Reihenfolge der Befehle .IF, .ELSE und .ENDIF stimmt und daß eine IF-ELSE-ENDIF-Konstruktion nicht geschachtelt werden darf.

Dieser Fehler tritt auch auf, falls vor einem .IF, .ELSE oder .ENDIF-Befehl ein Label steht. Dies ist nicht erlaubt.

Schreiben Sie das Label allein in eine freie Zeile direkt vor dem Befehl!

**.BASE MISSING ERROR**

Jeder Quelltext muß unbedingt eine Basis-Adresse enthalten. Definieren Sie diese am besten direkt zu Beginn des Quelltextes.

Es gibt eine Situation, in der der Assembler überfordert ist, und zwar dann, wenn Vorwärtsreferenzen auf Symbole erfolgen. Das bedeutet, daß ein Symbol verwendet wird, bevor es definiert wurde. Dieser Fall tritt dann ein, wenn ein Assembler-Befehl auf ein Symbol zugreift, aber dieses Symbol erst später im Quelltext definiert wird. In diesem Fall mußte eine Festlegung erfolgen, daß dieses zunächst unbekannte Symbol mit $8000 initialisiert wird. Ein

## Die Synchronisation

Assembler-Befehl, der eine Vorwärtsreferenz beinhaltet, wird also prinzipiell als 3-Byte-Befehl interpretiert. (Eine Ausnahme ist die Immediate-Adressierung. Hier ist klar, daß es sich um einen 2-Byte-Befehl handelt. Die Initialisierung erfolgt dann mit $80.) Zum Glück gibt es nur eine einzige Situation, in der dieser Fehler auftritt:

```
100.BASE $6000
110 <SPACE> LDA A
120 <SPACE> RTS
130.EQUATE A=198
```

Im ersten Pass nimmt der Assembler an, daß es sich in Zeile 110 bei LDA A um einen 3-Byte-Befehl handelt, denn er hat ja A=$8000 initialisiert. In Zeile 120 gilt dann * =$6003.

Im zweiten Pass nimmt der Assembler an, daß es sich in Zeile 120 bei LDA A um einen 2-Byte-Befehl handelt, denn er hat ja A=198, also eine Zero-Page-Adresse, in der Symboltabelle eingetragen. In Zeile 120 gilt dann * =$6002.

Diese Sitution, daß im ersten und im zweiten Pass unterschiedliche Adreßpegel ermittelt werden, nennt man einen Synchronisationsfehler. Der Assembler selbst merkt davon nichts, er erzeugt auch keine Fehlermeldung. Nur der Objectcode wird fehlerhaft, weil die Lücke von einem Byte bei der weiteren Assemblierung katastrophale Folgen hat.

Glücklicherweise tritt in Giga-Ass ein Synchronisationsfehler nur bei Vorwärtsreferenzen auf Zero-Page-Adressen auf, also wenn eine Zero-Page-Adresse erst hinter ihrer ersten Verwendung definiert wird. Man umgeht solche Vorwärtsreferenzen dadurch, daß man ein für allemal festlegt, daß Zero-Page-Adressen am Anfang des Quelltextes zu definieren sind. Dies sei Ihnen hiermit ans Herz gelegt und entspricht auch einem guten Programmierstil.

## Kompatibilität zum Hypra-Ass

Um für Giga-Ass eine Kompatibilität zum Hypra-Ass sicherzustellen, befindet sich auf der beiliegenden Diskette das Programm »HYPRA-KONVERT«. Eine Konvertierung ist aufgrund der Tokenverarbeitung von Giga-Ass notwendig, da Hypra-Ass alle Befehle und Mnemonics im Klartext auf Diskette speichert. Alle, die wissen wollen, wie dieses Konvertierungsprogramm arbeitet, finden unter »HYPRA-KONV.SRC« den dokumentierten Quelltext.

»INIT 31 K« verschiebt den Bildschirmspeicher des C64 nach $CC00. Dadurch steht Ihnen ein weiteres KByte für Quelltexte zur Verfügung. Den Quelltext dazu hat die Bezeichnung »SOURCE INIT 31K«.

Viel Spaß und Erfolg bei Ihren Programm-Projekten mit Giga-Ass. (Thomas Dachsel/gr)

# EIN LEISTUNGSFÄHIGER MONITOR

**Das kann »Promon 64«: Funktionen testen, ungewollte Abstürze vermeiden, Programme analysieren. Eine außergewöhnliche Hilfe für jeden Freund der Maschinensprache.**

Ein Assembler wie »Giga-Ass« gehört zu der Standardausrüstung jedes Programmierers. Er erlaubt Ihnen, eigene Maschinenprogramme zu erstellen. Doch jedem Programmierfreak ist es schon passiert, daß sein Werk aus zunächst undurchsichtigen Gründen abgestürzt ist. Das Überprüfen des Programmablaufs nach Schreibfehlern ist äußerst zeitintensiv und führt nicht immer zum Erfolg. Spätestens jetzt muß ein guter Monitor her. Erst mit Ihm wird es ohne großen Aufwand möglich, ein Programm durchzutesten.

Der Maschinensprachemonitor »Promon« überzeugt durch die Vollständigkeit seiner Befehle. Dadurch beträgt seine Länge allerdings stattliche 8 KByte. Da beim C64 im für Hilfsprogramme gern benutzten Speicherbereich ($C000 bis $CFFF) nur 4 KByte zur Verfügung stehen, wurde es nötig, dem »Promon« einen anderen Platz zu geben. Es bot sich der Bereich $8000 bis $A000 an, da ab hier bekanntlich der Modulstart des C64 beginnt. Damit wurde erreicht, daß Ihnen nach einem Reset sofort der Monitor zur Verfügung steht.

Zunächst eine kurze Beschreibung der im Artikel verwendeten Synonyme:

- (Start) = Beginn der Operation und ist eine 4stellige Hexadezimalzahl (»0000« bis »FFFF«)
- (Ende) = Ende einer Operation und ist ebenfalls eine 4stellige Hexadezimalzahl (»0000« bis »FFFF«)
- (Bytes) = Sind eines oder mehrere Bytes, mit denen die Funktion ausgeführt wird. Sie werden 2stellig eingegeben (»00« bis »FF«). Mehrere Bytes werden durch »Spaces« (Leerzeichen) getrennt.

Die Bildschirmausgabe kann zusätzlich auf den Drucker umgeleitet werden, wenn Sie die Befehle »geshiftet« (d.h. zusammen mit <SHIFT>) eingeben.

Sie laden »Promon 64« von der beiliegenden Diskette mit:

```
LOAD"PROMON-LADER",8
```

und starten mit RUN. Anschließend wird der eigentliche Monitor geladen (dieses Programm ist auf Diskette unter »PROMON 64« gespeichert) und kann auch mit »8,1« geladen werden.

**Starten des Monitors**

Das Programm startet nach dem Laden automatisch. Nach Verlassen des Monitors (siehe »X« und »N«) wird Promon durch Eingabe von »SYS 36864« und <RETURN> neu ge-

startet. Eine andere Methode ist das Drücken des RESET-Schalters (falls vorhanden) oder eine softwaremäßige Auslösung durch »SYS 64738« und <RETURN>.

### Monitorbefehle und Funktionen

**M(Start)(Ende) – Anzeige der Speicherinhalte**

Erlaubt das Anzeigen und Ändern des Speicherinhalts im Bereich von (Start) bis (Ende). Die Anzeige erfolgt zu jeweils acht Byte in einer Zeile und deren ASCII-Codes im Anschluß. Beispiel:

```
M20002200
```

zeigt den Speicherbereich von $2000 bis 2200. Nach der Bestätigung mit <RETURN> erscheint zunächst eine Zeile am Bildschirm. <SPACE> bringt die Darstellung zeilenweise auf den Bildschirm, <RUN/STOP> bricht die Ausgabe ab. Jede andere Taste erzeugt eine fortlaufende Anzeige bis zur Speicherstelle (Ende). Wird der Parameter (Ende) weggelassen, erfolgt die Ausgabe endlos. <RUN/STOP> bricht die Ausgabe ab.

**F(Start)(Ende)(Bytes) – Suchen nach Bytefolgen**

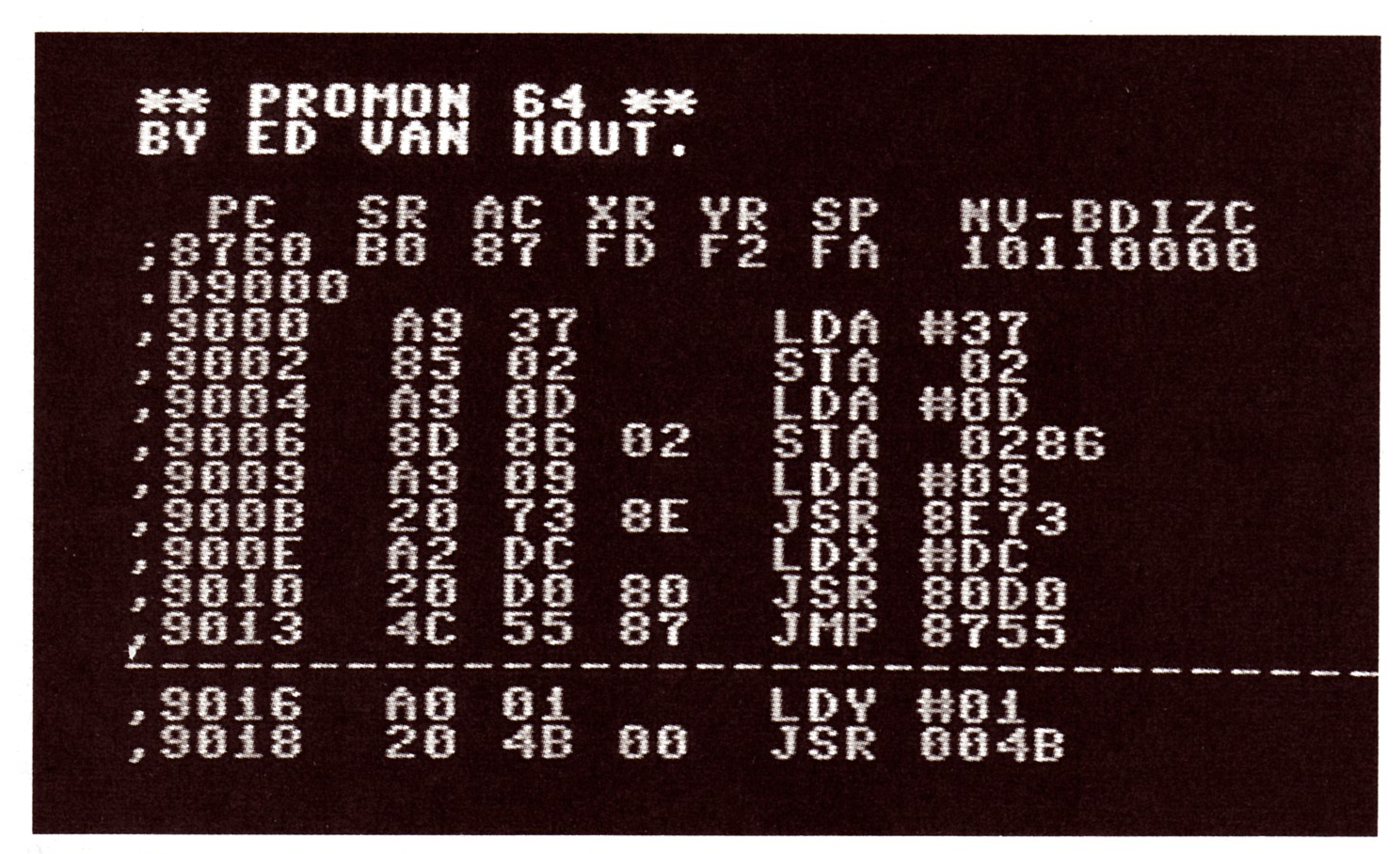



**Bild 1. Disassembliert ist hier der Programm-Einsprung in »Promon 64«**

Die Funktion »F« sucht im Speicher des C64 ab Adresse (Start) bis (Ende) nach der Bytefolge (Bytes). Werden die Bytes gefunden, so erscheint die Speicheradresse auf dem Bildschirm. Die Eingabe darf zwei Bildschirmzeilen nicht überschreiten (Leerzeichen dürfen weggelassen werden).

**R – Anzeige der Register**

Anzeige der aktuellen Registerinhalte. Diese Inhalte können durch Überschreiben geändert werden.

| PC | SR | AC | XR | YR | SP |
|---|---|---|---|---|---|
| Programm-zähler | Status-register | Akkumu-lator | X-Register | Y-Register | Stapel-zeiger |

Das Statusregister wird bitweise dargestellt (NV-BDIZC):
- N = Negative-Flag
- V = Overflow-Flag
-   = Unbelegt
- B = Break-Flag
- D = Dezimal-Flag
- I = Interrupt-Dissable-Flag
- Z = Zero-Flag
- C = Carry-Flag

**O(Start)(Ende)(Byte) – Füllen von Speicherbereichen**

Füllt den Speicherbereich von (Start) bis (Ende) mit dem Wert (Byte). Hierbei darf nur ein Byte als Argument angegeben werden. Beispiel:

```
O20002200FF
```

füllt den Bereich von $2000 bis $2200 mit dem Wert $FF.

**K(Start)(Ende) – Ausgabe im ASCII-Code**

Gibt den Speicherinhalt von (Start) bis (Ende) in ASCII-Codes aus. Wird (Ende) nicht angegeben, erfolgt eine fortlaufende Ausgabe ab (Start). Diese Funktion erlaubt das gezielte Suchen nach Texten im ASCII-Code.

**A(Start) – Assemblieren**

Die Assemblierung beginnt bei (Start). Die Befehle werden so eingegeben, wie sie der Disassembler zeigt. <RETURN> schließt die Zeile ab. Bei fehlerhafter Eingabe springt der Cursor wieder in die Anfangsposition zurück. Ansonsten wird der Befehl nach Ausgabe der Hex-Bytes gelistet. Falls Ihnen bei Sprüngen (Branch-Befehl, JSR und JMP) die Zieladressen noch nicht bekannt sind, geben Sie sogenannte »Labels« ein.

Ein Label besteht aus dem Buchstaben »M« (für Marke) und einer zweistelligen Hex-Zahl von 01 bis 30 (z.B.»BCC M01«). Wenn Sie die Zieladresse für diesen Sprung erreicht haben, dann kennzeichnen Sie diese mit der verwendeten Marke (beispielsweise: »M01 LDY #00«). Einzelne Bytes kennzeichnen Sie durch einen vorangestellten Punkt (z.B. ».00«).

Sie verlassen den Assembler durch Eingabe von »F« anstelle eines Befehls.

**G(Start) – Starten eines Maschinenprogramms**

Beginnt ein Maschinenprogramm ab der Adresse (Start).

Wird keine Adresse angegeben, so nimmt Promon den aktuellen Programmzähler (siehe »R – Anzeige der Register«).

**D(Start)(Ende) – Disassemblieren von Speicherbereichen**

Startet das Disassemblieren eines Speicherbereiches ab der Adresse (Start). (Ende) ist optional und muß nicht angegeben werden (Bild 1). Gibt man (Ende) nicht an, so wird die Anzeige durch Drücken einer beliebigen Taste angehalten und durch nochmaligen Tastendruck wieder gestartet.

**T(Start)(Byte) – Trace modus**

Startet ein Maschinenprogramm an der Adresse (Start) im »Trace«-Modus. Die Speicherkonfiguration des C64 (wie in Speicherstelle $01) ist dabei durch (Byte) definiert. Der Prozessor übernimmt in seine Register die Werte, die mit »R« abgerufen und manipuliert werden können. Der Opcode wird angezeigt, aber erst durch Drücken von <SPACE> ausgeführt. Anschließend erscheint der nächste Befehl. Trifft der Computer auf ein JSR mit Sprung ins Betriebssystem, so

## Kurzinfo: Promon 64

**Programmart:** Maschinensprache-Monitor
**Laden:** LOAD"PROMON-LADER",8
**Starten:** nach dem Laden RUN eingeben. Nach Verlassen des Monitors ist ein Neustart durch SYS 64738 oder durch einen Resettaster möglich.
**Steuerung:** über die Tastatur
**Besonderheiten:** Alle Anzeige-Funktionen werden zusätzlich über den Drucker ausgegeben, wenn man die entsprechenden Befehle mit gedrückter SHIFT-Taste eingibt.
**Benötigte Blocks:** 33 Blocks und 1 Block für das Ladeprogramm
**Programmautor:** Ed van Hout

kann dieser Aufruf durch Drücken von <J> durchgeführt werden. Danach wird vor dem nächsten Befehl wieder auf <SPACE> gewartet. Trace kann durch <RUN/STOP> angehalten werden. Beim Erreichen eines BRK- oder RTS- Befehls stoppt das Programm automatisch.

**X – Verlassen des Monitors**
Rückkehr ins Basic. Die Modulkennung des Promon 64 wird dadurch nicht zerstört. Der Monitor kann durch ein »Reset« jederzeit aufgerufen werden.

### Umrechen- und Rechenfunktionen

**#(Dezimalzahl) – Umrechnung Dezimal in Hexadezimal**
Rechnet eine (Dezimalzahl) in hexadezimal um. Falls der Wert kleiner als 256 ist, wird zusätzlich in Binär umgewandelt.

**$(Hexadezimalzahl) – Umrechnung Hexadezimal in Dezimal**
Rechnet eine (Hexadezimalzahl) in dezimal um. Falls der Wert kleiner als $0100 ist, wird zusätzlich in Binär umgewandelt.

**%(Binärzahl) – Umrechnung Binär in Hexadezimal und Dezimal**
Rechnet eine achtstellige Binärzahl (z.B. %00000011) in den entsprechenden Hexadezimal- und Dezimalwert um.

**? – Rechnen im Promon**
Erlaubt das Rechnen im Monitor. Die einzelnen Zahlen der drei Zahlensysteme Hexadezimal, Dezimal oder Binär können addiert oder subtrahiert werden. Dabei ist es zulässig die einzelnen Systeme zu mischen. Beispielsweise »?#23+$10+%00000001« ergibt »$001E 00011110 #30«.

### Vergleichs- und Verschiebe-Befehle

**=(Start alt)(Ende alt)(Start neu) – Vergleichen eines Speicherbereichs**
Vergleicht den Speicherbereich von (Start alt) bis (Ende alt) mit dem Bereich ab (Start neu). Werden unterschiedliche Speicherstellen festgestellt, so wird deren Adresse ausgegeben. Ein Beispiel:

```
=1000 2000 3000
```

Vergleicht den Speicherbereich $2000 bis $2000 mit $3000 bis $4000.

**W(Start alt)(Ende alt)(Start neu) – Verschieben ohne Umrechnung**
Verschiebt einen Speicherbereich von (Start alt) bis (Ende alt) nach (Start neu). Dabei wird keine Adreßumrechnung für Sprungbefehle vorgenommen. Das bedeutet, der neue Speicherbereich ist mit dem alten identisch. Sinnvoll ist diese Funktion für die Verschiebung von reinen Daten.

**V(Start alt)(Ende alt)(Start neu)(Start)(Ende) – Umrechnung ohne Verschieben.**
Rechnet alle Sprungadressen im Bereich (Start) bis (Ende) um. Es findet keine Verschiebung des Programms statt. Als Kriterium für die Umrechnung wird der Bereich (Start alt) bis (Ende alt) bezüglich der neuen Adresse (Start neu) genommen. Haben Sie ein Programm an eine andere als die vorgesehen Adresse geladen, hilft Ihnen diese Funktion das Programm lauffähig zu machen.

**C(Start alt)(Ende alt)(Start neu)(Start)(Ende) – Umrechnung mit Verschieben.**
Verschiebt einen Speicherbereich von (Start alt) bis (Ende alt) nach (Start neu). Dabei werden alle Sprungadressen im Bereich von (Start) bis (Ende) an den neuen Speicherbereich angepaßt. Diese Funktion gibt Ihnen die Möglichkeit ein Programm an einen anderen Speicherplatz zu verschieben und dort in lauffähiger Form zu testen. Es steht anschließend - wie für diesen Speicherbereich geschrieben - zur Verfügung.

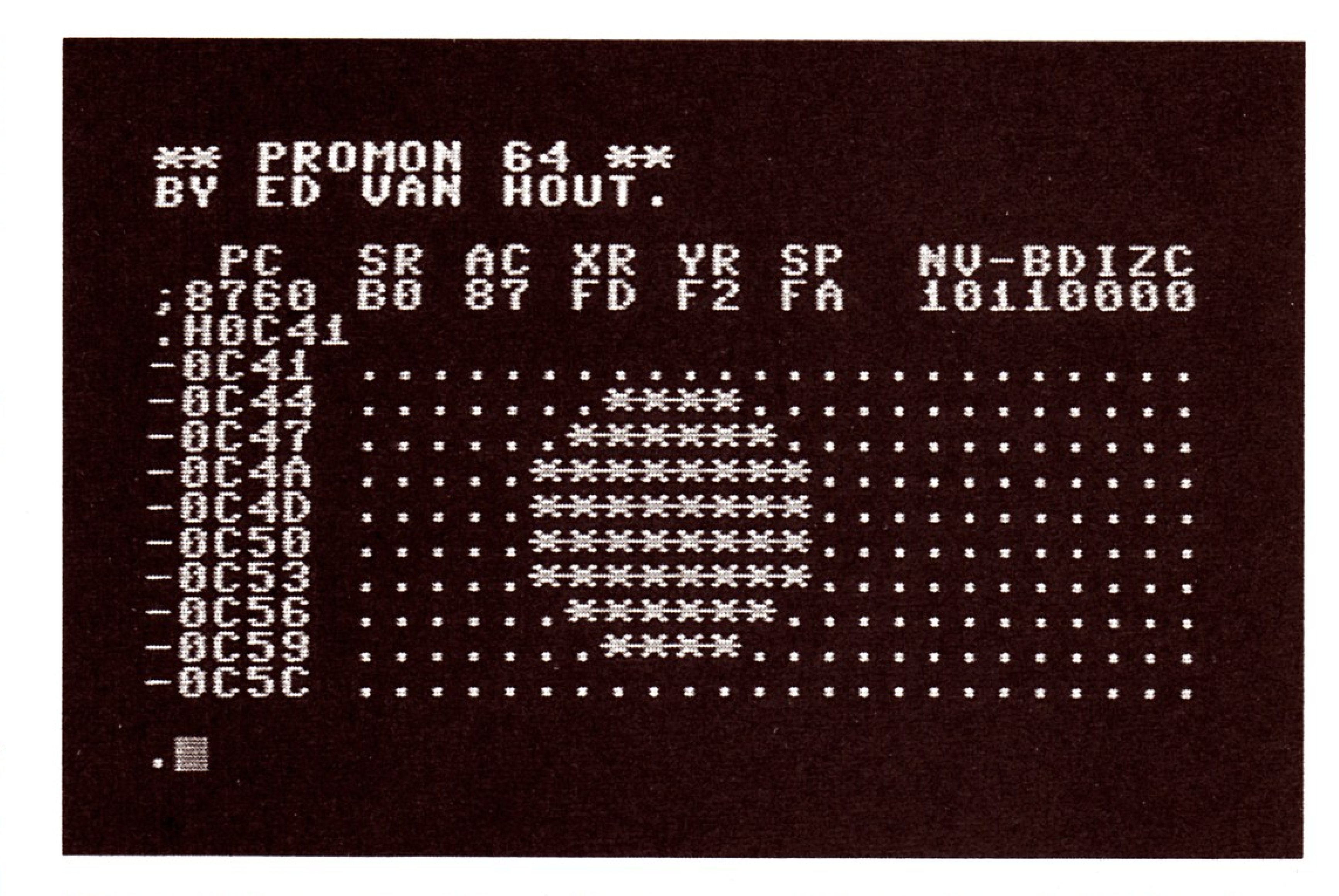



Bild 2. Mit der Funktion »H« lassen sich Sprites leicht finden

### Sonderfunktionen für Grafik, Sprites und Bildschirm

**J – Suchen von Grafikbildern**
Erlaubt das Suchen von Grafikbildern im Speicher des Computers. Dabei wird mit den Tasten <1> bis <8> der gewünschte Speicherbereich angewählt. Die Tasten <H> und <M> erlauben das Umschalten zwischen Hires- und Multicolor-Darstellung. Durch die Funktionstaste <F1> ändern Sie die Rahmenfarbe. <F3> und <F5> funktionieren nur in Multicolor und ändern dabei zwei Farben der Darstellung.

**H(Start)(Ende) – Suchen nach Sprites**
Zeigt den Speicherbereich des Computers von (Start) bis (Ende) bitweise an (Bild 2). Dabei werden immer drei Byte nebeneinander als Bitmuster angezeigt (Bit gesetzt = »*«, Bit gelöscht = ».«). Das Auffinden von Sprites wird dadurch erleichtert. Editieren ist durch einfaches Überschreiben und <RETURN> möglich. (Ende) ist optional und muß nicht angegeben werden.

**U(Start)(Ende) – Anzeige im Bildschirmcode**
Gibt ab der Adresse (Start) alle Speicherinhalte in Bildschirmcodes aus. Dabei werden keine Adressen ausgegeben, um den Bildschirmaufbau nicht zu zerstören. Beim Abbruch der Funktion durch <RUN/STOP> wird die gerade erreichte Speicheradresse ausgegeben.

### Befehle zum Editieren und Manipulieren eines Speicherbereiches

**E(Adresse) – Editieren im Bildschirmcode**
Mit diesem Befehl wird der Speicherbereich ab (Adresse) über die Tastatur mit Zeichen editiert. Diese Zeichen werden nach Drücken von <F3> im entsprechenden Speicherbereich als Bildschirmcode abgelegt. <F7> bringt Sie zurück in den Befehlseingabe-Modus.

**QE(Start)(Ende)(Byte) – Exclusiv-Oder-Verknüpfung (EOR)**
Dieser Befehl führt im Bereich von (Start) bis (Ende) ein EOR mit dem angegebenen Wert (Byte) aus und erlaubt dadurch das Codieren größerer Speicherbereiche.

**QA(Start)(Ende)(Byte) – Addition**
Dieser Befehl addiert von (Start) bis (Ende) den Wert (Byte) zu den einzelnen Speicherstellen.

### Befehle zur Basic-Bearbeitung

**N – »Old«-Funktion**
Holt ein Basic-Programm, das mit NEW gelöscht wurde zurück. Anschließend wird der Promon 64 verlassen und ins Basic gesprungen.

**B(Start)(Ende) – Data Zeilen Erstellen**
Dieser Befehl erzeugt Basic-DATA-Zeilen aus dem Speicherbereich von (Start) bis (Ende). Die dazu benötigte Zeit wird nach <RETURN> angezeigt.

### Eingabe und Ausgabeoperationen

**I(Gerätenummer) – Einstellung der Geräteadresse**
Stellt die Standard-Gerätenummer für alle Disketten-Operationen auf den mit (Gerätenummer) bezeichneten Wert. Voreingestellt ist »8«.

**S″Name″(Start)(Ende) – Speicherfunktion**
Speichert einen Bereich von (Start) bis (Ende) auf das durch »I« voreingestellte Gerät.

**L″Name″(Start) – Ladefunktion**
Lädt ein Programm von einem Gerät, das mit »I« bestimmt wurde. (Start) ist optional und dient dazu, ab dem unter (Start) bestimmten Speicherbereich zu laden. Beispiel:

```
L"TEST"4000
```

lädt das Programm »TEST« an die Speicherstelle $4000. Lassen Sie die Adressenangabe (4000) fort, wird das Programm an die Originaladresse geladen.

**(Start) – Suchfunktion für Datasette**
Nur für Besitzer einer Datasette: Dieser Befehl sucht auf der Kassette nach einem Programm, das mit Turbo-Tape gespeichert wurde. Wird ein solches Programm gefunden, so wird dessen Name angezeigt. Durch Drücken von <SPACE> wird das Programm geladen. Ein <RUN/STOP> bricht diesen Vorgang ab.

**P(Gerätenummer) – Umstellung der Druckeradresse**
Setzt die Ausgabeadresse für Druckfunktionen auf den unter (Gerätenummer) bestimmten Wert.

**<SHIFT> – Ausgabe auf dem Drucker**
Die Bildschirmdarstellungen lassen sich zusätzlich auf einen angeschlossenen Drucker ausgeben. Drücken Sie dazu den Befehl zusammen mit <SHIFT>.

### Befehle des eingebauten Disketten-Monitors

**ZR(Track)(Sektor) – Lesen von Track und Sektor in den Speicher**
Liest (Track) und (Sektor) von der Diskette in den Speicher des C64. Läßt man die Parameter weg, wird der zuletzt angegebene Sektor nochmals gelesen.

**ZW(Track)(Sektor) – Schreiben von Track und Sektor auf Diskette**
Schreibt den Speicherinhalt auf den (Track) und (Sektor) der Diskette zurück. Läßt man die Parameter weg, wird auf den zuletzt angegebenen Sektor geschrieben.

**ZN – Einlesen des logisch nächsten Sektors**
Liest den nächsten Sektor eines Programms von Diskette. Dies dient zum Verfolgen von files.

**Z – Ausgabe des zuletzt angesprochenen Disketten-Sektors**
Gibt die Spur- und Sektornummer des zuletzt von Diskette angesprochenen Sektors aus. Wurde der Disketten-Monitor noch nicht verwendet, erfolgt keine Ausgabe.

**Z(Monitorkommandos) – Verwendung der Monitorkommandos auf Diskette**
Alle Monitorkommandos des Promons können auf den Speicherblock des Diskettenmonitors bezogen verwendet werden. »ZK« zeigt beispielsweise alle Bytes als ASCII-Zeichen.

»Z$« zeigt den im Disketten-Monitor verwendeten Speicherbereich an.

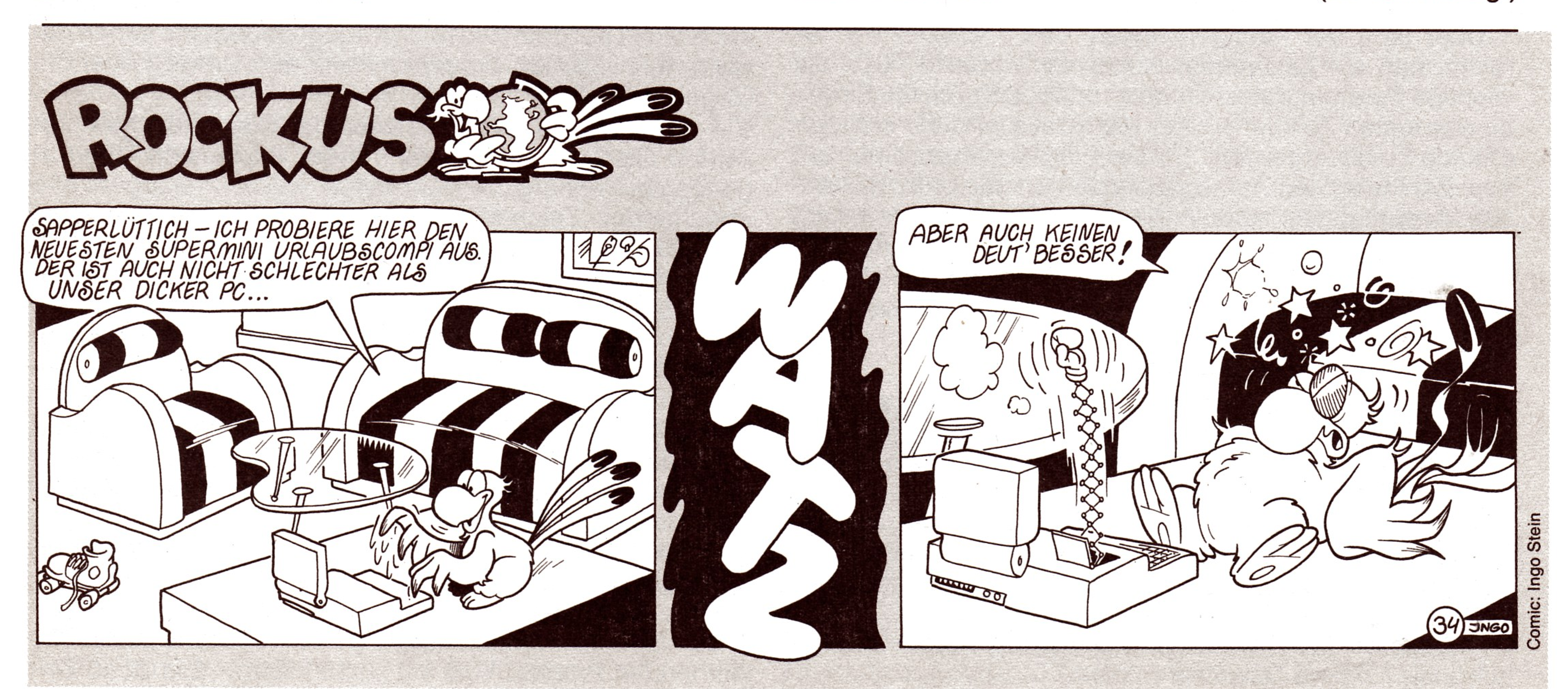
**@(DOS 5.1) – Befehle zum Diskettenlaufwerk**
Hier können die Befehle zur Diskettenstation übermittelt werden. Die Syntax entspricht dabei der des DOS 5.1 (siehe Test/Demo-Diskette, die jeder Floppystation beim Kauf beiliegt). Das Inhaltsverzeichnis rufen Sie beispielsweise mit »@$« auf.

### Sonderbefehle des Promon 64

**Y – Anzeige der Speicheraufteilung**
Zeigt an, ob der Monitor im RAM oder im ROM arbeitet.

**YA – Umschaltung auf RAM**
Schaltet den Monitor auf RAM-Betrieb. Finden jetzt Operationen statt, die im Bereich des Basic-ROM, des I/O-Bereichs oder des Betriebssystem-ROM liegen, wird generell der »darunterliegende« RAM-Bereich angesprochen.

**YO – Umschaltung auf ROM**
Schaltet den Monitor auf ROM-Betrieb. Es wird möglich, das Basic-Rom oder den KERNEL-Bereich zu betrachten. Änderungen dieser Speicherinhalte sind natürlich nicht möglich. Der Bereich $D000 bis $DFFF wird umgeschaltet auf den I/O-Bereich. (Ed van Hout/gr)






**New York, 23.55 Uhr – die neuesten Informationen aus Übersee. Auch das ist möglich mit »Proterm V6.0«, dem hervorragenden Terminalprogramm für den C64.**

Ob Sie mit Mailboxen im Inland oder Ausland in Verbindung treten wollen, dieses Datenfernübertragungs-Programm (DFÜ-Programm) der Spitzenklasse besitzt alle Eigenschaften, die für den Anwender von Bedeutung sind. Der C64 ist zwar durch seinen 40-Zeichen-Bildschirm etwas benachteiligt, wenn es um die Datenfernübertragung geht. Dieses Manko gleicht Proterm V6.0 jedoch durch eine komfortable Benutzerführung und großen Komfort aus.

Es spielt keine Rolle, ob Sie einen Akustikkoppler oder ein Modem besitzen. Proterm V6.0 unterstützt beide Gerätekonfigurationen. Zusätzlich zum Übertragungsgerät benötigen Sie Ihren C64 mit einer Floppy 1541 und eventuell einen Drucker.

Sie laden das Programm von der beiliegenden Diskette mit

```
LOAD "PROTERM V6.0",8
```

und starten es mit RUN. Folgende Diskettendateien werden nach dem Programmstart nachgeladen:
- PRO.KEYS
- PRO.TEL

Falls sich diese Files nicht auf der Diskette befinden, gibt das Programm eine Fehlermeldung aus und geht nach Tastendruck zur Hauptfunktion über. Wie die genannten Dateien auf Diskette angelegt werden, erfahren Sie später in dieser Beschreibung.

Übertragen Sie oft größere Texte oder sogar Programme, dann sind für Sie die beiden eingebauten Editoren interessant. Sie erlauben das unabhängige Bearbeiten zweier verschiedener Dateien, wobei auch das Senden einer Datei direkt von der Diskette möglich ist. Empfangen Sie Daten zum Beispiel aus einer Mailbox, die für Sie von Wichtigkeit sind, so können Sie einen Drucker mitprotokollieren lassen oder den gesamten Text entweder in einen Zusatzpuffer oder direkt auf eine Diskette schreiben.

Eine der herausragenden Fähigkeiten von Proterm V6.0 ist der eingebaute Scanner, der das Suchen von Datex-P-NUAs zum Kinderspiel werden läßt. Hierbei können Sie Parameter eingeben, die eine gezielte Suche zulassen und nicht nur das Finden jedes möglichen Anschlusses erlauben. Eine sehr nützliche Einrichtung, die dank ihres durchdachten Konzepts schnell unentbehrlich wird.

Neben den eben aufgezählten Eigenschaften von Proterm V6.0 darf man natürlich auch das XModem-Protokoll für die Datenübertragung nicht vergessen. Hier können Sie Dateien von einer Diskette schnell und bequem senden oder von einem anderen Computer empfangen. Dabei spielt das Datenformat keine Rolle. Es werden sowohl Programm- als auch sequentielle Dateien behandelt.

Über jede der Funktionstasten <F1> bis <F8> erreicht man ein bestimmtes Menü von Proterm V6.0, das wichtige Parametereinstellungen erlaubt. Durch die Eingabe des ersten Zeichens einer Menüzeile wird der entsprechende Punkt angewählt. Bei der Eingabe kann es sich um eine Zahl oder den Anfangsbuchstaben des entsprechenden Menüpunktes handeln. Nach dem Tastendruck erfolgt eine direkte Reaktion des Programms oder eine Abfrage auf weitere Parameter in der Kopfzeile des Bildschirms. Wir wollen uns nun die einzelnen Menüpunkte etwas genauer ansehen.

**<F1> — Parameter einstellen:**

0 — ASCII: on/off. Hier wird der intern und auf Diskette verwendete Zeichencode eingestellt. In Stellung »on« wird ASCII-Code benutzt, in Stellung »off« arbeitet Proterm V6.0 mit den Commodore-Codes.

1 — Übertragungsrate: 300/600/1200 Bit/s. Unter diesem Punkt können Sie die Übertragungsgeschwindigkeit einstellen. Ein Druck auf die Taste <1> schaltet jeweils zwischen den drei möglichen Werten um.

2 — Databits: 7/8. Hier können Sie zwischen einer Übertragung mit 7 oder 8 Datenbits wählen.

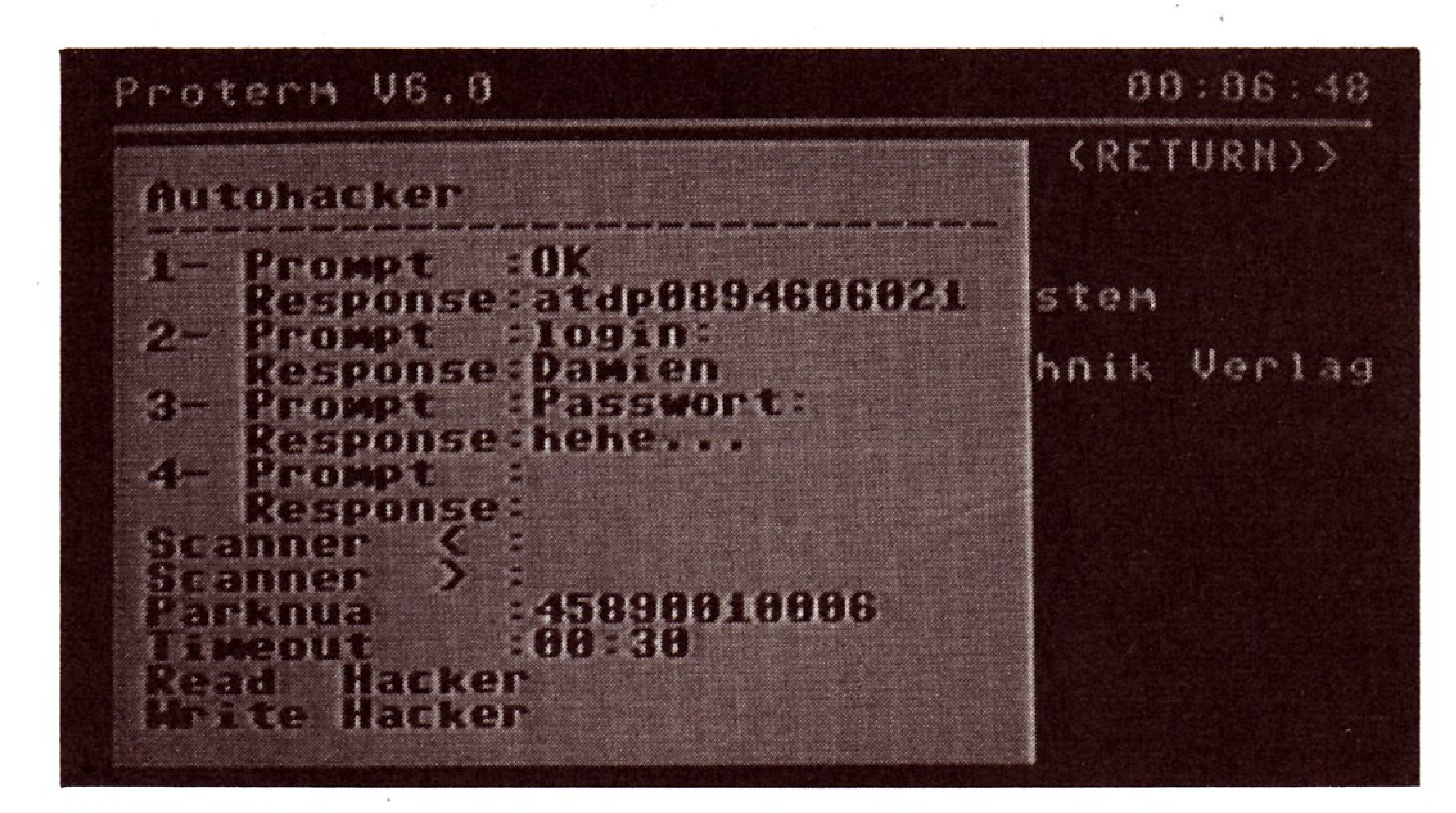



**Bild 1. Mit dem Autohacker suchen Sie Anschluß**

3 — Stop-Bits: 1/2. Die Anzahl der Stop-Bits stellen Sie mit der Taste <3> ein.

4 — Parity: No/Mark/Space/Even/Odd. Die Art der Parität (Prüfbits) können Sie unter Punkt 4 einstellen.

5 — Duplex: Full/Half/Host. Übertragungsart einstellen. In der Einstellung »Host« werden alle Zeichen, die Sie senden auch direkt auf dem Bildschirm ausgegeben. Sie sehen dadurch Ihre eigenen Eingaben ohne den Umweg über die Gegenstelle.

6 — Tempo: Fast/Slow. Einstellen der Sendegeschwindigkeit. In Stellung »Fast« wird mit der maximalen Übertragungsgeschwindigkeit gearbeitet; »Slow« verzögert alle abgehenden Zeichen um einen fest eingestellten Wert. Diese Funktion wird sinnvoll, wenn eine Gegenstelle Zeichen verschluckt, weil ihre Verarbeitungsgeschwindigkeit zu gering ist.

7 — Linefeed: Off/On. In der Stellung »On« wird jedem ankommenden Carriage Return ($0D) automatisch ein Linefeed ($0A) angehängt.

8 — Printer: CBM/ASCII. Hier wird die Druckerausgabe auf CBM- oder ASCII-Code eingestellt, wobei jeder Nicht-Commodore-Drucker ohne Interface mit der Einstellung »ASCII« angesteuert werden sollte.

9 — Sec.adr.: 7/0/1/2. Wahl der Sekundäradresse für einen angeschlossenen Drucker.

**<F2> — Funktionstasten:**

Die Tastenkombinationen <CTRL 1> bis <CTRL 5> können bei Proterm V6.0 mit kurzen Texten (bis zu 16 Zei-

chen) belegt werden. Beim Druck auf die jeweilige Tastenkombination wird dann der String mit abschließendem $0D ausgegeben.

Im Funktionstastenmenü existiert zusätzlich die Möglichkeit, die Tastenbelegung mit »Save Control-Keys« unter dem Dateinamen »PRO.KEYS« auf eine Diskette zu speichern. Bei jedem Neustart von Proterm V6.0 wird die entsprechende Belegung dann automatisch nachgeladen und steht wieder zur Verfügung. Voraussetzung dazu ist allerdings, daß sich die Diskette mit der entsprechenden Datei vor dem Start von Proterm V6.0 im Laufwerk befindet.

**<F3> — Textspeicher 1, <F4> — Textspeicher 2:**

Bei Proterm V6.0 haben Sie neben dem Bildschirm, der normalerweise sichtbar ist, noch zwei Textspeicher mit je 2000 Byte Größe, die unabhängig voneinander bearbeitet werden können. Das Arbeiten mit einem Textspeicher wird durch die Taste <F3> für den Textspeicher 1 und durch <F4> für den Textspeicher 2 ermöglicht.

<E> — Edit page: Umschalten in den Editor des jeweiligen Textspeichers. Die betreffende Seite kann hier, wie vom Basic-Editor gewohnt, editiert werden. Spezielle, nur innerhalb des Editors erreichbare, Befehle sind:

<F2> — Leerzeile einfügen
<F4> — ganze Zeile löschen
<SHIFT RETURN> — Rest der Zeile löschen
<HOME> — Cursor an den Textanfang
<CLR> — Cursor an das Textende
<F3> — Editor verlassen

<L> — Load Page: Laden der betreffenden Textseite von einer Diskette. Nach dem Druck auf die Taste <L> wird nach dem Dateinamen gefragt, anschließend werden die Daten geladen.

<S> — Save Page: Hier wird die betreffende Textseite nach der Angabe eines gültigen Dateinamens auf eine Diskette gespeichert.

<T> — Transmit: Die gewählte Textseite wird als ASCII-Code mit den eingestellten Übertragungsparametern gesendet. Unterbrechen können Sie die Übertragung mit der Taste <RUN/STOP>; Wollen Sie danach mit dem Senden fortfahren, drücken Sie einfach eine beliebige Taste. Das Beenden erfolgt mit <RUN/STOP> und anschließendem Druck auf die Taste <C>.

<K> — Kill page: Die aktuelle Textseite wird durch Druck auf die Taste <K> im Speicher gelöscht.

Jeder Druck auf eine beliebige andere Taste außer den oben angeführten bringt Sie wieder in den Hauptbildschirm von Proterm.

**<F5> — Textpuffer:**

Alle ankommenden Zeichen können parallel zur Ausgabe auf dem Bildschirm in einen, 29854 Zeichen fassenden, internen Puffer geleitet und dort gespeichert werden. Mit <↑> wird der Puffer geöffnet und mit <↑> auch wieder geschlossen. Bei geöffnetem Puffer erscheint der Programmname in der Statuszeile revers.

<L> — List Buffer: Der Inhalt des Puffers wird ausgegeben. Die Anzeige wird mit <RUN/STOP> angehalten und mit einer beliebigen Taste fortgesetzt oder mit der Taste <C> ganz abgebrochen.

<P> — Print Buffer: Der Inhalt des Puffers wird auf einem angeschlossenen Drucker ausgedruckt.

<K> — Kill Buffer: Der Pufferinhalt wird gelöscht.

<S> — Save Buffer: Der Inhalt des Puffers wird auf Diskette gespeichert, nachdem ein Dateiname angegeben wurde.

»Free«: Hinter diesem Text wird die noch verbleibende Aufnahmekapazität des Puffers angezeigt.

**<F6> — Diskfiles:**

<L> — List File: Eine Datei von der Diskette wird auf dem Bildschirm ausgegeben. Die Anzeige wird mit <RUN/STOP> angehalten und mit einer beliebigen Taste fortgesetzt oder mit <C> ganz abgebrochen. Ein Trick: Wurde vorher der Puffer geöffnet, so wird jetzt auch parallel in den Puffer geschrieben. Hiermit kann man kleine Dateien aneinanderhängen.

<C> — Command: Unter diesem Menüpunkt können Kommandos an das angeschlossene Diskettenlaufwerk gesendet werden. Die Rückmeldung der Floppystation wird auf dem Bildschirm ausgegeben.

<T> — Transmit: Eine Datei wird direkt von der Diskette gelesen und gleichzeitig in der ASCII-Norm gesendet. Anhalten können Sie diesen Vorgang mit <RUN/STOP>; fortfahren mit beliebiger Taste. Ein Abruch erfolgt mit <RUN/STOP> und dann <C>.

<D> — Directory: Ausgabe des Inhaltsverzeichnisses einer eingelegten Diskette.

**<F8> — Übertragung einer Datei per XModem:**

<T> — Transmit File: Eine Datei wird von der Diskette geladen und gleichzeitig im XModem-Protokoll übertragen. Sollen Programmdateien gesendet werden, ist nur der Filename anzugeben. Bei sequentiellen Dateien ist dem Dateinamen ein »,S« anzuhängen. XModem überträgt Blöcke zu je 128 Byte. Nach zehn Fehlversuchen wird abgebrochen. Mit <CTRL X> wird die Übertragung nach dem nächsten richtig übertragenen Block abgebrochen.

<R> — Receive File: Eine Datei wird im XModem-Protokoll empfangen und direkt auf eine Diskette gespeichert. Für die Tastenfunktionen gelten die Angaben unter »Transmit«.

## Komfortables Bearbeiten von Dateien

»Blocks«: Hinter diesem Text erfolgt Ausgabe der Nummer des aktuellen Blocks, der gerade übertragen wird.

»Errors«: Angabe der Fehlversuche, auf den aktuellen Block bezogen.

Proterm V6.0 bietet neben den Menüfunktionen noch einige Sonderzeichen zur Steuerung bestimmter Funktionen an. Die Aufgabe dieser Zeichen wurde zum Teil schon deutlich oder wird weiter unten näher erklärt. Hier zunächst eine Liste aller Sonderzeichen:

<CBM U> — setzt die Uhr auf Null
<↑> — schaltet den RAM-Puffer ein und aus
<↑> (verwendet in beliebigem Text) — das jeweils folgende Zeichen wird als CTRL-Code ausgegeben.
<SHIFT ↑> — »Autohacker« aktivieren
<←> — Store
<*> — Jokerzeichen für Password
<£> — Scannerjoker
<@> — Parken

**Der »Autohacker«:**

Proterm V6.0 bietet die Möglichkeit, im Autohacker vier verschiedene Prompt/Response-Sequenzen zu definieren. Bei exakter Übereinstimmung der eingehenden Zeichen mit einem Prompt-String (Erkennungsmeldung), der frei vorgegeben werden kann, wird der dazugehörige Response-String (Antwortmeldung) automatisch abgeschickt. Wurde keine

### Kurzinfo: Proterm V6.0

**Programmart:** Datenfernübertragung (DFÜ)
**Laden:** LOAD "PROTERM V6.0",8
**Starten:** nach dem Laden RUN eingeben
**Steuerung:** Tastatur
**Benötigte Blocks:** 47 Blocks + mindestens 1 Block für die Datenfiles »PRO.KEYS« und »PRO.TEL«
**Programmautor:** K.P. Steenken

Antwortmeldung definiert, so erfolgt nur die Ausgabe eines »Carriage Return«. Auf diese Weise ist es möglich, mit dem Autohacker einen bestimmten Datex-P-Anschluß zu suchen und auf dessen Erkennungsmeldung zu reagieren.

Die Reaktion auf eine bestimmte Erkennungsmeldung (im weiteren Verlauf als »Response« bezeichnet) wird bei entsprechender Programmierung automatisch von Proterm V6.0 verändert. Um dies zu erreichen, werden in den Response-String Platzhalter (Joker) eingebaut.

<*> — Kennwort-Joker: Anstatt des <*> werden von Proterm V6.0 »Paßworte« aus einer Tabelle im Textspeicher zwei gesendet. Diese »Paßworte« stehen im Textspeicher 2 durch Kommas getrennt.

| Mailbox-Nr. | Name | Parameter | Online | Sysop |
|---|---|---|---|---|
| 0812141477 | AL CAPONE-Box | 7/N/1 | 24 h | Andy |
| 081451307 | HUNTBOX | 8/N/1 | 24 h | Klaus |
| 0308034656 | The System | 8/N/1 | 24 h | Andreas |
| 02151476567 | KBCK 2, Krefeld | 8/N/1 | 24 h | Karl |
| 089152670 | A.C.M-Box | 8/N/1 | 24 h | (Name des Sysops stand bei Redaktionsschluß noch nicht fest) |

**Tabelle 1. Eine Auswahl von Mailboxen, die Sie rund um die Uhr erreichen können**

<£> — Scanner-Joker: An den Stellen, die mit dem Scanner-Joker versehen sind, werden vom Scanner in Proterm V6.0 Texte generiert und gesendet. Diese Texte hängen vom Eintrag bei »Scanner<« und »Scanner>« ab. Der Startstring »Scanner<« wird zeichenweise alphabetisch so lange erhöht, bis der Endstring »Scanner>« erreicht ist; dazu ein Beispiel:
Scanner< = AAA
Scanner> = CCC
Ausgabe: AAA, AAB, AAC, ABA, ABB, ABC, ACA, ACB, ACC, BAA, AAB, BAC, BBA, BBB, BBC, BCA, BCB, BCC, CAA, CAB, CAC, CBA, CBB, CBC, CCA, CCB, CCC

Im Scanner können alle druckbaren ASCII-Zeichen verwendet werden.

Neben den Jokern benutzt der Autohacker noch einige Sonderzeichen:
<↑> — Control: Das dem <↑> folgende Zeichen wird als CTRL-Code ausgegeben.

## Scannen mit dem C 64

<←> — Store: Dieses Zeichen kann sowohl direkt, als auch in einem Text verwendet werden. Der zuletzt ausgegebene, links oben in der Statuszeile stehende String wird in Textspeicher eins geschrieben.
<@> — Parken: Beim Auffinden dieses Zeichens wird, wenn ein Timeout abgelaufen ist, eine Park-NUA angewählt. Der Rest von Response wird dann überlesen.

NUA ist die Abkürzung von »Network-User-Adress« (Netzwerk-Benutzer-Adresse).
Ein Beispiel für das Arbeiten mit dem Autohacker:
Ziel: R-NUAs scannen, im Bereich R-45400040000 bis R-45400040099 und gefundene NUAs speichern
Datex-P meldet sich mit: »DATEX-P 123456789«.

Wird die Verbindung hergestellt, meldet Datex-P: »Verbindung... ... (128)«
Als Eintrag im Autohacker (Bild 2) (Einschalten nicht vergessen!) schreiben wir:
1- Prompt : 56789
Response: @r 454000400£
2- Prompt : 128)
Response: ←↑pclr
Scanner< : 00
Scanner> : 99
Parknua : »Park-NUA«
Timeout : 00:45

Nach Beendigung des Scannens, stehen alle gefundenen NUAs in Textspeicher eins.

**Der »Autodialer«:**
Achtung: Der Autodialer arbeitet mit dem Textspeicher 1 zusammen. Autodialer und Autohacker würden sich gegenseitig stören. Deshalb ist ein Wählen bei eingeschaltetem Autohacker nicht möglich.

Der Autodialer wird mit <CBM D> aktiviert. Sie haben die Möglichkeit, den ganzen Textspeicher 1 mit Telefonnummern zu beschreiben. Die Nummern müssen immer als erstes eingegeben werden. Danach können Sie noch diverse wichtige Erläuterungen anfügen (zum Beispiel Paßwörter, Login-Zeiten, Dateinamen für Autologon etc.). Wenn Sie den Textspeicher 1 nach der Eingabe sämtlicher Telefonnummern mit dem Dateinamen »PRO.TEL« auf eine Diskette speichern, werden die Telefonnummern nach einem Neustart von Proterm V6.0 automatisch nachgeladen. Das funktioniert allerdings nur dann, wenn sich die entsprechende Diskette im Laufwerk befindet, bevor Sie den Befehl RUN eingeben.

Es wäre ratsam, mit 40 Zeichen pro Zeile auszukommen, da sonst nicht der ganze Text in der Kopfzeile erscheint. Falls der Textspeicher leer sein sollte, macht der Computer darauf aufmerksam (»Page Empty«). Wenn Sie nun im Autodial-Modus sind, können Sie mit den Cursortasten (<CURSOR-aufwärts> und <CURSOR-abwärts>) die Nummernliste scrollen. Anschließend haben Sie folgende Möglichkeiten:
<F1> — die ausgesuchte Nummer wird gewählt.
<F3> — die ausgesuchte Nummer wird so lange gewählt, bis ein Carrier empfangen wird. (Abbruch mit <CTRL X>)
<F5> — alle Nummern im Textspeicher werden nacheinander gewählt, bis sich ein Anschluß mit Carrier meldet (Abbruch mit <CTRL X>). Nach Beendigung des Wählvorganges geht der Computer, sofern ein Carrier vorhanden ist, automatisch in den Terminal-Modus zurück. Man kann den Autodial-Modus jederzeit mit <RUN/STOP> verlassen. Wenn eine Verbindung beendet werden soll und erneut <CBM D> gedrückt wird (wieder bei ausgeschaltetem Autohacker), fragt der Computer, ob aufgelegt werden soll. Die Frage ist nur mit <RETURN> zu bestätigen, ansonsten gelangt man in den Terminalmodus, ohne daß die Verbindung abgebrochen wurde.

Das Sonderzeichen »↑« bewirkt eine zusätzliche Verzögerungszeit von ungefähr vier Sekunden im Autodial-Modus.

Die Funktion des Autodialers ist an das Resco-Modem angepaßt. Besitzer anderer Modems werden diese komfortable Wähl-Methode leider nicht nutzen können.

**Autologon**

Wird in der Nummernliste hinter einer Rufnummer ein Dateiname in Anführungszeichen angegeben ("Dateiname"), so lädt Proterm V6.0 nach erfolgreichem Wählvorgang den Autohacker mit der Datei nach und schaltet dann den Autohacker und den Terminal-Modus ein.

Damit sind wir am Ende der Anleitung zu Proterm V6.0 angekommen. Es wird sicher eine ganze Weile dauern, bis Sie an die Grenzen dieses hervorragenden Terminalprogramms für den C 64 stoßen. Bis dahin wünschen wir Ihnen viel Spaß bei Ihren Entdeckungen in der großen Welt der Datenfernübertragung.

(K.P. Steenken/bl)



# Der Weg zum richtigen Ton

**Die fantastischen Fähigkeiten des C64 zur Klangerzeugung sind allgemein bekannt. Doch wissen Sie, wie man dem Computer die tollsten Töne und Musikstücke entlockt? Lernen Sie das »Sound-Wunder« in der folgenden Einführung gründlich kennen.**

Zur Erzeugung von Geräuschen und Musik ist der C64 mit einem leistungsfähigen Baustein ausgestattet. Er trägt die Bezeichnung 6581 und soll hier im folgenden SID genannt werden. SID steht für »Sound Interface Device«, was man mit »Klang-Schnittstellen-Baustein« übersetzen könnte. Der SID ist eigentlich ein kleiner Synthesizer, der dreistimmige Melodien spielen oder drei unabhängige Geräusche gleichzeitig erzeugen kann oder auch eine Kombination von beiden, zum Beispiel eine zweistimmige Melodie oder ein Geräusch. Wie man ihn dafür programmiert, soll hier gezeigt werden. Da das Standard-Basic des C64 keine speziellen Befehle zu diesem Zweck vorsieht, muß man sich näher mit dem inneren Aufbau des SID befassen, um ihn dann mit PEEK- und POKE-Befehlen zu steuern. Dieser gezwungenermaßen etwas unelegante Programmierstil hat aber wenigstens einen Vorteil für denjenigen, der in Maschinensprache programmieren kann oder es lernen will. Er kann nämlich die PEEK- und POKE-Befehle direkt in die Assemblersprache übernehmen. Stürzen wir uns also gleich mitten hinein in die SID-Programmierung (in Basic).

Beim SID wird ein Klang durch folgende Parameter (= Steuergrößen) beeinflußt:

| | |
|---|---|
| 1. Lautstärke | |
| 2. Hüllkurve | Sie steuert den zeitlichen Lautstärkenverlauf zum Beispiel eines ausklingenden Tones. |
| 3. Kurvenform | Sie ist für den Klangcharakter des Tones verantwortlich. |
| 4. Frequenz | Sie entspricht der Tonhöhe. |

Mit Einzelheiten und mit weiteren Parametern zur Klangsteuerung werden wir uns gleich befassen. Zunächst wollen wir aber einmal einen Ton erzeugen, zum Beispiel um zu hören, ob unser Monitor oder Fernseher, der die Töne wiedergeben muß, richtig eingestellt ist (Perfektionisten schließen den C64 über die Audio/Video-Buchse und ein normales DIN-Überspielkabel an die Hi-Fi-Anlage an). Folgende Pokes helfen bei dieser Einstellung:

| | |
|---|---|
| POKE 54296,15 | stellt den SID auf maximale Lautstärke |
| POKE 54278,240 | wählt eine einfache Hüllkurve. |
| POKE 54273,67 | stellt eine Frequenz ein (zirka 1000 Hz). |
| POKE 54276,17 | wählt eine sogenannte Dreieckskurve und schaltet zugleich den Ton ein (muß immer als Letztes geschehen!). |

Jetzt müßte ein Ton hörbar sein, der ähnlich wie bei einem Fernseh-Testbild klingt.

| | |
|---|---|
| POKE 54276,16 | schaltet den Ton wieder ab. |

Der gleiche Ton mit schärferem Klang gefällig?

| | |
|---|---|
| POKE 54276,33 | wählt eine »Sägezahnkurve«. Diese klingt heller und schärfer als das Dreieck. |

Doch anstatt mit geheimnisvollen POKEs zu arbeiten, sollten wir uns zuerst systematisch mit dem SID befassen. Wer jedoch nur einen Klangeffekt für ein eigenes Programm benötigt und an den Einzelheiten des Sound-Chips nicht interessiert ist, kann diesen analytischen Teil überspringen und ab Kapitel »Klangvolle Effekte« weiterlesen. Dort finden Sie die Erläuterungen zu den Listings 1 bis 4.

Unter einem Register versteht man in der Computertechnik einen Speicherplatz, der mit einer besonderen Funktion gekoppelt ist. Diese Speicherplätze sind also nicht dazu da, um Daten darin abzulegen, sondern um eine Funktion auszulösen oder um Informationen über den Zustand eines Bausteins zu bekommen. Man unterscheidet demnach Schreibregister und Leseregister.

Der SID verfügt insgesamt über 25 Schreib- und Leseregister. Auf Bild 1 sind diese in grafischer Form dargestellt. Der SID hat die Basisadresse:
S = 54272 (dezimal) oder $D400 (hexadezimal)

Unter dieser und den 28 folgenden Adressen können die Register des SID angesprochen werden. Wir werden in Zukunft Registeradressen wie in Bild 1 immer in der Form S+n (n = 0 bis 28) angeben, weil diese Schreibweise prägnanter als eine fünfstellige Zahl ist. Es ist empfehlenswert, sich auch in Programmen an diese Vereinbarung zu halten.

Das Registerschema gliedert sich in drei Blöcke:
Der erste Block ist in Wirklichkeit dreimal vorhanden, für jede Stimme einmal. Die sieben Register dieser Blöcke haben also für die drei Stimmen unterschiedliche Adressen, wie links im Schema auch angegeben ist.

Der zweite Block (S+21 bis S+24) dient hauptsächlich zur zusätzlichen Klangbeeinflussung durch einen Filter. Den Filter werden wir aber erst später behandeln. Aus diesem Block interessiert zunächst nur die rechte Hälfte des Registers S+24, das für die Lautstärke zuständig ist.

Der dritte Block (S+25 bis S+28) besteht aus vier sogenannten »Nur-Lese-Registern«. Aus diesen Registern kann nur gelesen werden, Schreibzugriffe bleiben wirkungslos. Auch diese Register, die Spezialeffekten dienen, interessieren uns zunächst noch nicht.

Ein Register besteht, wie jeder andere Speicherplatz beim C64 auch, aus einem Byte, beziehungsweise 8 Bit. Man sieht, daß einige Register noch in Felder unterteilt sind. Bei diesen Registern haben einzelne Bits oder Bitgruppen unterschiedliche Bedeutung. Die schraffierten Bereiche kennzeichnen Bits, die keine Funktion im SID haben. Wir werden bald sehen, wie man einzelne Bits innerhalb eines Byte gezielt ansprechen kann. Nun zu den Registern im einzelnen: Es werden beim ersten Block stellvertretend die Register der Stimme 1 (S+0 bis S+6) beschrieben. Die Register für Stimme 2 (S+7 bis S+13) und Stimme 3 (S+14 bis S+20) sind in ihrer Funktion identisch.

Ab hier ist es praktisch, wenn man bei der Lektüre das kleine Programm aus Listing 1 im Computer hat, denn dann kann man die Wirkung der Parameter in den SID-Registern gleich ausprobieren. Die Parameter stehen gut les- und editierbar in den DATA-Zeilen. Das Programm erzeugt nach dem Starten einen Ton bei einem beliebigen Tastendruck. Tasten mit Auto-Repeat-Funktion, wie zum Beispiel die Space-Taste, erzeugen einen Dauerton. Abgebrochen wird das Programm mit der RUN/STOP-Taste. Der letzte Parameter steuert übrigens die Tonlänge durch eine einfache Verzögerungsschleife.

## Frequenz S+0 und S+1

Die Frequenz kann beim SID auf 16 Bit genau angegeben werden. Eine 16-Bit-Zahl kann Werte zwischen 0 und 65535 annehmen. Dieser Wert entspricht allerdings nicht der Frequenz in Hz (Hertz = Schwingungen pro Sekunde). Der SID-Wert F zu einer gegebenen Frequenz in Hz errechnet sich nach:

```
F = 17.0284 * Frequenz
```

Der SID-Wert F zum sogenannten Kammerton a mit 440 Hz beträgt also (ganzzahlig gerundet):

```
F = 17.0284 * 440  ≈  7492
```

Die höchste vom SID erzeugbare Frequenz beträgt dann (gerundet):

```
65535 / 17.0284  ≈  3849 (Hz)
```

Zum Experimentieren mit Klangeffekten interessiert uns die genaue Frequenz eigentlich gar nicht, für korrekt gestimmte Tonleitern müssen wir sie dagegen kennen. Zunächst wollen wir aber erfahren, wie man den SID mit dem Wert F (Frequenz) programmiert. Diese im Dezimalsystem maximal fünfstellige Zahl wird im Binärsystem durch 16 Bit dargestellt. Da es sich beim C64 um einen 8-Bit-Mikrocomputer handelt, müssen wir diesen Wert in zwei 8-Bit-Hälften, das sogenannte niederwertige und höherwertige Byte, kurz Low-Byte und High-Byte zerlegen. Hier zwei »Rezepte«:

1. Methode (Standard):

```
HI = INT(F/256)
LO = F-256*HI
```

Das ist nichts anderes als eine Division durch 256 mit Rest. HI ist dabei der Quotient und LO der Divisions-

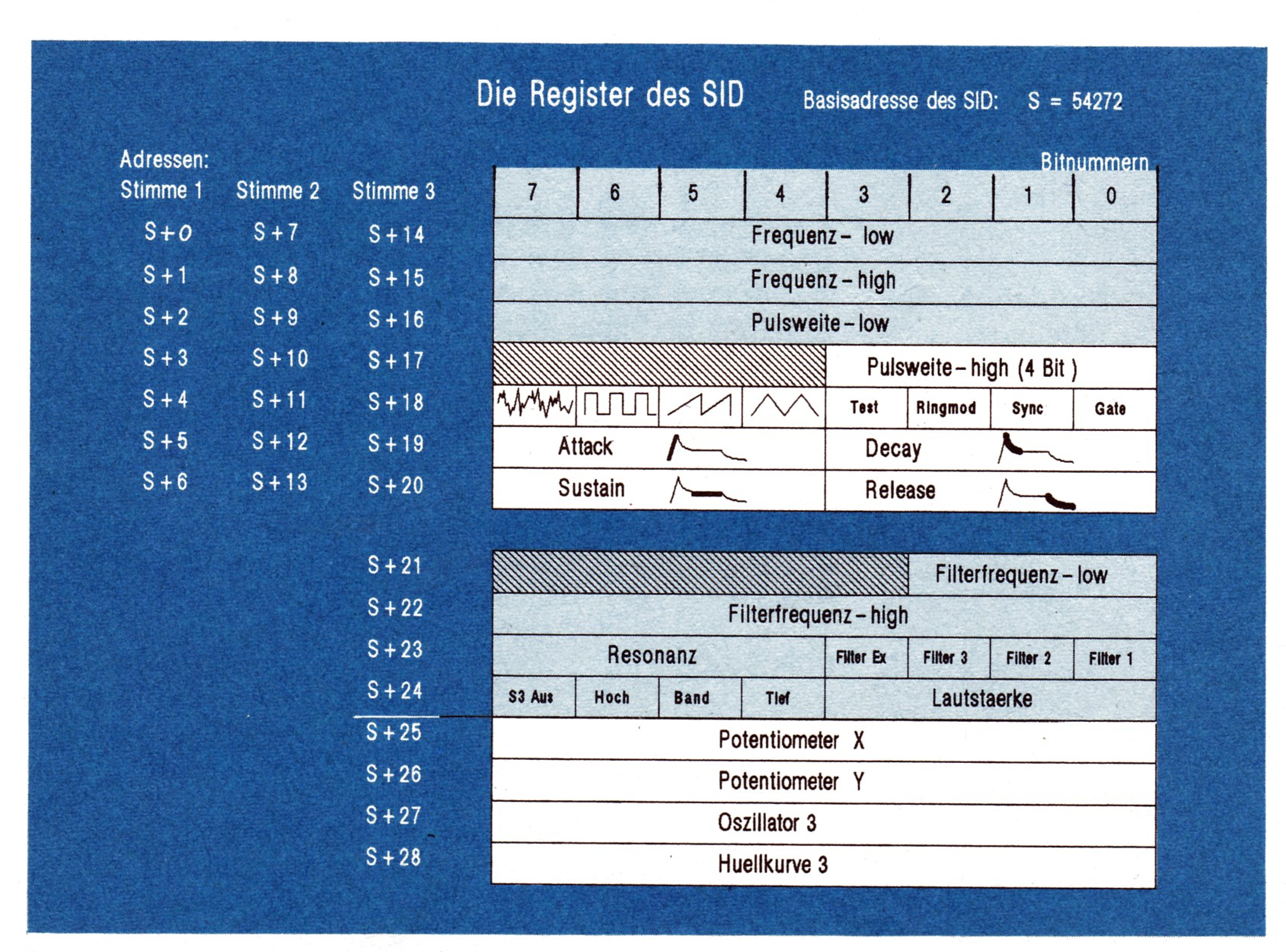



**Bild 1. Alle Register des Sound-Chip auf einen Blick**

rest. Die Werte LO und HI sind beides Byte-Werte und liegen damit im Bereich 0 bis 255. Im Fall F=7492 (entsprechend 440 Hz) ergibt sich zum Beispiel:

```
HI = 29 und LO = 68
```

2. Methode (mit Einschränkungen, aber schneller):

```
HI = F/256
LO = F AND 255
```

Die INT-Funktion zur Berechnung von HI wurde hier gespart. HI kann hier noch Nachkommastellen haben; diese werden aber später von dem noch folgenden POKE-Befehl abgeschnitten. Die Berechnung von LO funktioniert hier nur bei F-Werten im Bereich 0 bis 32767. Die zweite Methode ist nur dann zu empfehlen, wenn es auf Geschwindigkeit ankommt.

Mit den Werten LO und HI müssen wir dann die beiden Register S+0 und S+1 besetzen:

```
POKE S+0,LO
POKE S+1,HI
```

Man kann die Wirkungsweise des High- und Low-Bytes auch als Grob- und Feineinstellung auffassen. Oft genügt für einen Klang eine grobe Frequenzsteuerung. Man braucht dann nur das High-Byte zu berücksichtigen und kann das Low-Byte ein für allemal zum Beispiel auf 0 setzen.

## Pulsweite S+2 und S+3

Der Parameter »Pulsweite« ist nur wirksam, wenn als Kurvenform das Rechteck gewählt wurde. Die Kurvenformen sind in Bild 1 bei Register S+4 grafisch dargestellt und werden im nächsten Abschnitt besprochen. Das Rechteck ist eine Kurvenform, die nur zwischen zwei Werten hin- und herspringt. Ist der obere Wert genauso lang wie der untere, so spricht man von einer symmetrischen Rechteckkurve. Das Verhältnis zwischen der Länge des oberen und des unteren Wertes kann mit dem Parameter »Pulsweite«, im folgenden P genannt, gesteuert werden. P kann Werte von 0 bis 4095 annehmen und wirkt sich auf die Klangfarbe des Tones aus. Das symmetrische Rechteck, das man mit P = 2048 erhält, klingt verhältnismäßig hohl und wird als typischer Rechteckklang bezeichnet. Entfernt man sich mit P von 2048 in Richtung 0 oder 4095, so wird der Klang zunehmend heller und später schnarrend oder zirpend. Maßgeblich ist hierbei nur der Abstand von P zum Mittelwert 2048. So klingt zum Beispiel P = 2048+500 genauso wie P = 2048–500. Bei P=0 und P=4095 wird kein Ton mehr erzeugt.

P ist eine 12-Bit-Größe und muß wie F in ein Low- und ein High-Byte zerlegt werden. Beim High-Byte können dabei nur die unteren vier Bit gesetzt sein. Zu diesem Zweck kann man ohne Einschränkungen die schon beschriebene Methode 2 anwenden:

```
HI = P/256
LO = P AND 255
POKE S+2,LO
POKE S+3,HI
```

## Steuerregister S+4

Dieses Register ist für mehrere Funktionen gleichzeitig zuständig:
- Die Wahl der Kurvenform
- Ein- und Ausschalten des Tones
- Spezialeffekte Ringmodulation und Synchronisation
- Reset der Stimme

Zunächst einmal eine Tabelle mit den Funktionen im einzelnen:

| Bit | Dezimalwert (POKE...) | Funktion |
|---|---|---|
| 0 | 1 | GATE schaltet Ton ein und aus |
| 1 | 2 | SYNC Synchronisation (Spezialeffekt) |
| 2 | 4 | RING Ringmodulation (Spezialeffekt) |
| 3 | 8 | TEST Reset |
| 4 | 16 | wählt Dreieckskurve |
| 5 | 32 | wählt Sägezahnkurve |
| 6 | 64 | wählt Rechteckkurve |
| 7 | 128 | wählt Rauschen |

Mit einem POKE an die Adresse S+4 werden immer alle acht Bit gleichzeitig beeinflußt. Einen Befehl zum Setzen oder Löschen einzelner Bits gibt es nicht. Man muß sich daher über die gewünschten Werte aller acht Bits im klaren sein, auch wenn man nur ein Bit verändern will. Um den richtigen POKE-Wert zu erhalten, müssen die Wertigkeiten der Bits, die man setzen will, addiert werden. Die folgenden drei Beispiele sollen zur Veranschaulichung dienen:

1. Rechteck wählen und Ton einschalten
   Bits: 6 und 0
   = Byte-Wert: 2↑6 + 1 = 65
   POKE S+4,65
2. Ton abschalten, Rechteck gewählt lassen
   Bits: 6
   Byte-Wert: 2↑6 = 64
   POKE S+4,64

Anmerkung: Beim Abschalten eines Tons sollte man immer die zuletzt gewählte Kurvenform gewählt lassen, damit der Ton ausklingen kann. Mit POKE S+4,0 (alle Bits rücksetzen) wird der Ton abrupt abgebrochen.

3. Dreieck mit Ringmodulation wählen, Ton einschalten
   Bits: 4, 2 und 0
   Byte-Wert: 2↑4 + 2↑2 + 2↑0 = 16 + 4 + 1 = 21
   POKE S+4,21

## Die Kurvenform

Sie bestimmt die Klangfarbe des Tones. Am vielseitigsten ist das schon besprochene Rechteck, weil man es durch die Pulsweite reichhaltig gestalten kann. Der Sägezahn klingt noch etwas heller und strahlender als das Rechteck. Er eignet sich besonders gut zur Imitation mancher Instrumentenklänge wie Streicher und Blechbläser. Das Dreieck klingt dagegen weich und dumpf und ist bei tiefen Tönen leider leise. Der Klang ist aber bei hohen Tönen sehr angenehm. Rauschen eignet sich für Effekte wie Wind, Düsenlärm, Schüsse, Explosionen und Schlagzeugklänge. Das Rauschen hat zwar keine feste Tonhöhe, doch sein Klangcharakter wird durch den Frequenzparameter entscheidend beeinflußt.

Indem man zwei Kurvenform-Bits gleichzeitig setzt, kann man durch Kombination mehrerer Kurvenformen weitere Klänge erzeugen. Dabei werden die Einzelklänge nicht etwa einfach gemischt, sondern andere Kurvenformen erzeugt. Rauschen läßt sich allerdings nicht mit einer anderen Kurvenform kombinieren. Auch die Kombination von drei Kurvenformen ist unbrauchbar. Sie liefert einen leisen, fast im Rauschen untergehenden Klang. Es bleiben also drei Kombinationen, die sehr interessant klingen:

| **Wellenform** | **POKE-Wert (An/Aus)** |
|---|---|
| Rechteck - Sägezahn | 97/96 |
| Rechteck - Dreieck | 81/80 |
| Sägezahn - Dreieck | 49/48 |

Der Klangcharakter variiert stark von tiefen zu hohen Tönen. Die letzte Kombination liefert nur bei sehr tiefen Tönen gute Resultate. Der Klang der ersten beiden Kombinationen hängt natürlich auch von der Pulsweite P ab.

## Die Spezialeffekte

Sie sollen hier nur am Rande erwähnt werden. Wird das SYNC-Bit für Stimme 1 gesetzt (Bit 1 in Register S+4), so kann Stimme 1 nicht mehr frei schwingen, sondern wird von Stimme 3 mit beeinflußt, man sagt hier »synchronisiert«. Auch das Ring-Bit bewirkt, daß Stimme 3 die Stimme 1 beeinflußt. Diese sogenannte Ringmodulation wirkt allerdings nur auf die Dreieckskurve. Der Effekt ist daher nur hörbar, wenn das Ring-Bit (Bit 2) zusammen mit Bit 4 für Dreieck gesetzt wird. Beide Effekte liefern ähnliche Resultate. Es lassen sich unter anderem metallische und glockenähnliche Klänge erzeugen. Die Stimme 3 braucht dabei nicht über ihr GATE-Bit eingeschaltet werden. Maßgeblich ist nur die Frequenz von Stimme 3 (Register S+14 und S+15).

Nun besitzen natürlich auch Stimme 2 und 3 je ein SYNC- und ein RING-Bit. Die drei Stimmen steuern sich dabei nach dem Schema:

Stimme 1 ⟶ Stimme 2
Stimme 2 ⟶ Stimme 3
Stimme 3 ⟶ Stimme 1

Das TEST-Bit wird man wahrscheinlich nie benötigen. Es übt eine lokale Reset-Funktion auf die jeweilige Stimme aus. Solange es gesetzt ist, ist nichts hörbar, unabhängig von den anderen Bits. Wenn man allerdings versucht, Rauschen mit einer anderen Kurvenform zu kombinieren, kann es passieren, daß die betroffene Stimme gewissermaßen »abstürzt« und nichts mehr von sich gibt. Man kann sie dann mit einem gezielten Reset über das TEST-Bit wieder zum Leben erwecken.

## Die Hüllkurven S+5 und S+6

Wenn eine Stimme über das GATE-Bit eingeschaltet wird, dann folgt ihr zeitlicher Lautstärkenverlauf einer programmierbaren Hüllkurve. Die Hüllkurve bestimmt unter anderem, ob der Ton hart oder weich einsetzt und ob er schnell oder langsam ausklingt. Der Name kommt von den vier Phasen, die die Hüllkurve durchläuft. Jeder Phase ist dabei ein Parameter zugeordnet.

**Attack** Die Attack-Phase wird durch das Setzen des GATE-Bits eingeleitet. Der Pegel steigt dabei von 0 bis Maximum (Lautstärkeregister) an. Die Zeit für diesen Anstieg ist über den Parameter A in 16 nicht-linearen Stufen von 2 ms bis 8 s einstellbar. Eine kurze Attack-Phase bewirkt einen unmittelbaren und harten Toneinsatz wie bei Schlag- oder Zupfinstrumenten. Eine mittlere Attack-Zeit ist typisch für Bläser- und Streicherklänge, und mit einer langen Attack-Zeit kann man einen Ton wie am Mischpult langsam einblenden.

**Decay** Nachdem der Maximalwert erreicht ist, fällt der Pegel in der Decay-Phase bis auf den Sustain-Pegel ab. Die Zeit dazu ist mit dem Parameter D in 16 nicht-linearen Stufen von 6 ms bis 24 s einstellbar.

**Sustain** nennt man die Phase nach dem Pegelabfall in der Decay-Phase. Der Ton klingt dann so lange auf dem Sustain-Pegel weiter, bis das GATE-Bit zurückgesetzt wird. Der Parameter SU bestimmt hier also keine Zeit, sondern einen Pegel, und zwar in 16 linearen Stufen von Null bis Maximum.

**Release** Beim Rücksetzen des GATE-Bits wird der Ton nicht einfach abgeschaltet, sondern nimmt in der Release-Phase gleichmäßig vom Sustain-Pegel bis nach Null ab. Die Zeit dazu ist in der gleichen Abstufung wie die Release-Zeit über den Parameter R einstellbar.

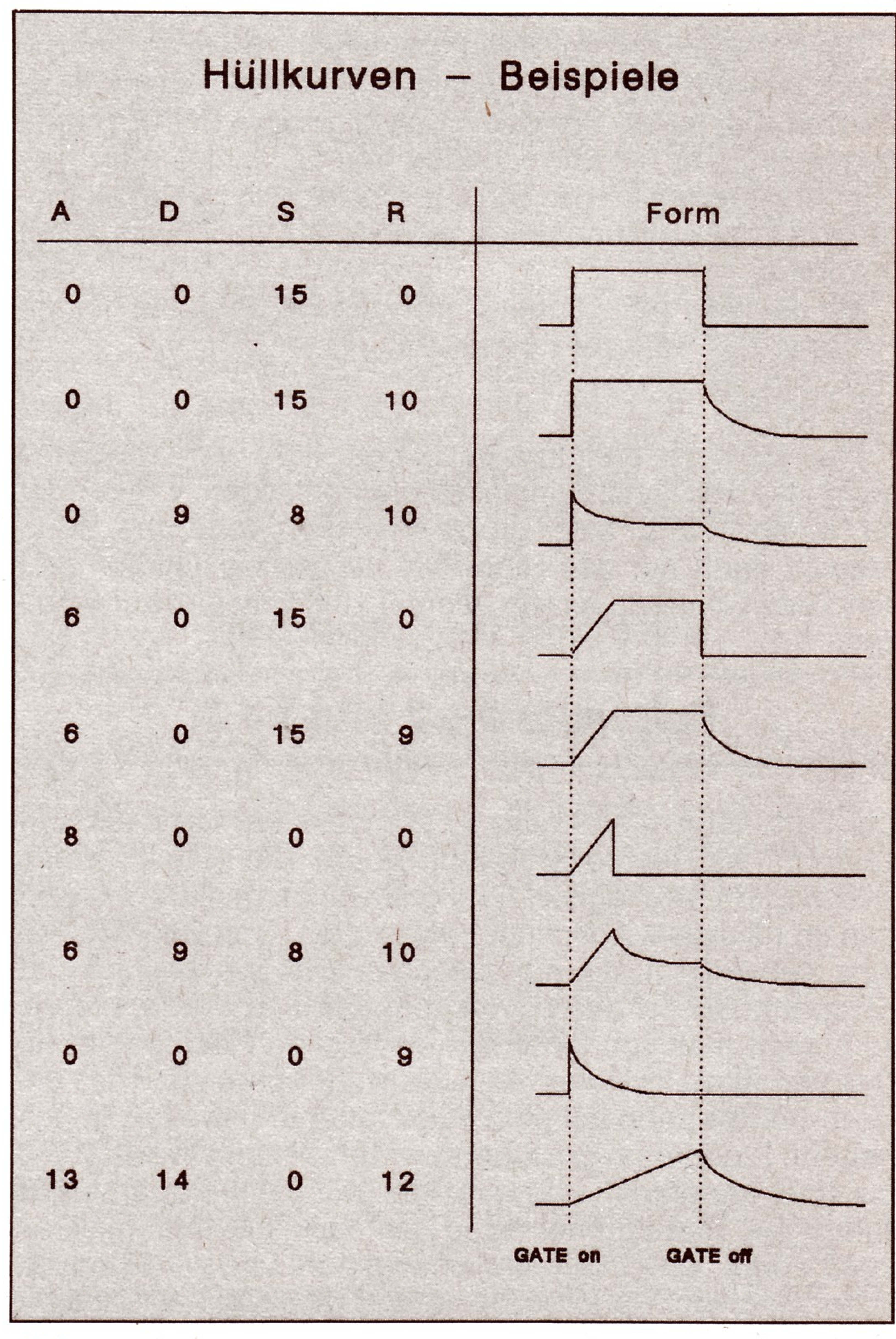



**Bild 2. Hier sehen Sie, wie sich unterschiedliche Werte für ADSR auf die Hüllkurve auswirken**

Wenn das GATE-Bit bereits vor Erreichen der Sustain-Phase rückgesetzt wird, dann startet die Release-Phase mit dem aktuellen Pegel der Hüllkurve. Auf diese Weise ergeben sich:

a) Attack-Decay-Release-Zyklen ADR oder gar nur
b) Attack-Release-Zyklen AR

Bei einem Sustainpegel von Null sind Decay und Release funktionell gleichwertig (9. Beispiel in Bild 2). Bei einem maximalen Sustain-Pegel entfällt die Decay-Phase (Attack-Sustain-Release-Zyklus ASR, 1. Beispiel in Bild 2).

## Sonderfälle

Die Parameter A, D, SU und R sind 4-Bit-Werte. Jeweils zwei von ihnen werden wie folgt in ein Register gepackt:

```
POKE S+5,16*A+D
POKE S+6,16*SU+R
```

Bild 3 zeigt einige Hüllkurvenbeispiele.

Die bisher erworbenen Kenntnisse befähigen Sie, eindrucksvolle Klänge aus dem SID herauszulocken. Probieren Sie die verschiedenen Möglichkeiten der Klangerzeugung aus. Was un noch fehlt, ist die weiterführende Beeinflussung der Töne, beispielsweise durch Filter.

Der Filter bietet neben der Wahl der Kurvenform eine weitere Möglichkeit zur Klangbeeinflussung. Im Gegensatz zu den vorher beschriebenen Parametern muß man sich aber nicht um den Filter kümmern, da er abschaltbar ist und weil der SID auch ohne Filter reichhaltige Klänge erzeugen

## Filter, Register S+21 bis S+24

kann. Um die Wirkungsweise eines Filters zu verstehen, muß man sich einen Klang aus mehreren sogenannten Sinustönen zusammengesetzt denken, dem Grundton und den Obertönen. Sinustöne sind gewissermaßen die nicht mehr weiter zerlegbaren Atome in der Akustik. Ein reiner Sinuston klingt dumpf und ohne charakteristische Färbung. Die vom SID erzeugte Dreiecksschwingung kommt vom Klang her einem Sinuston recht nahe. Die anderen Kurvenformen verdanken ihren helleren Klang einem reichhaltigeren Obertonspektrum (= Folge von Obertönen).

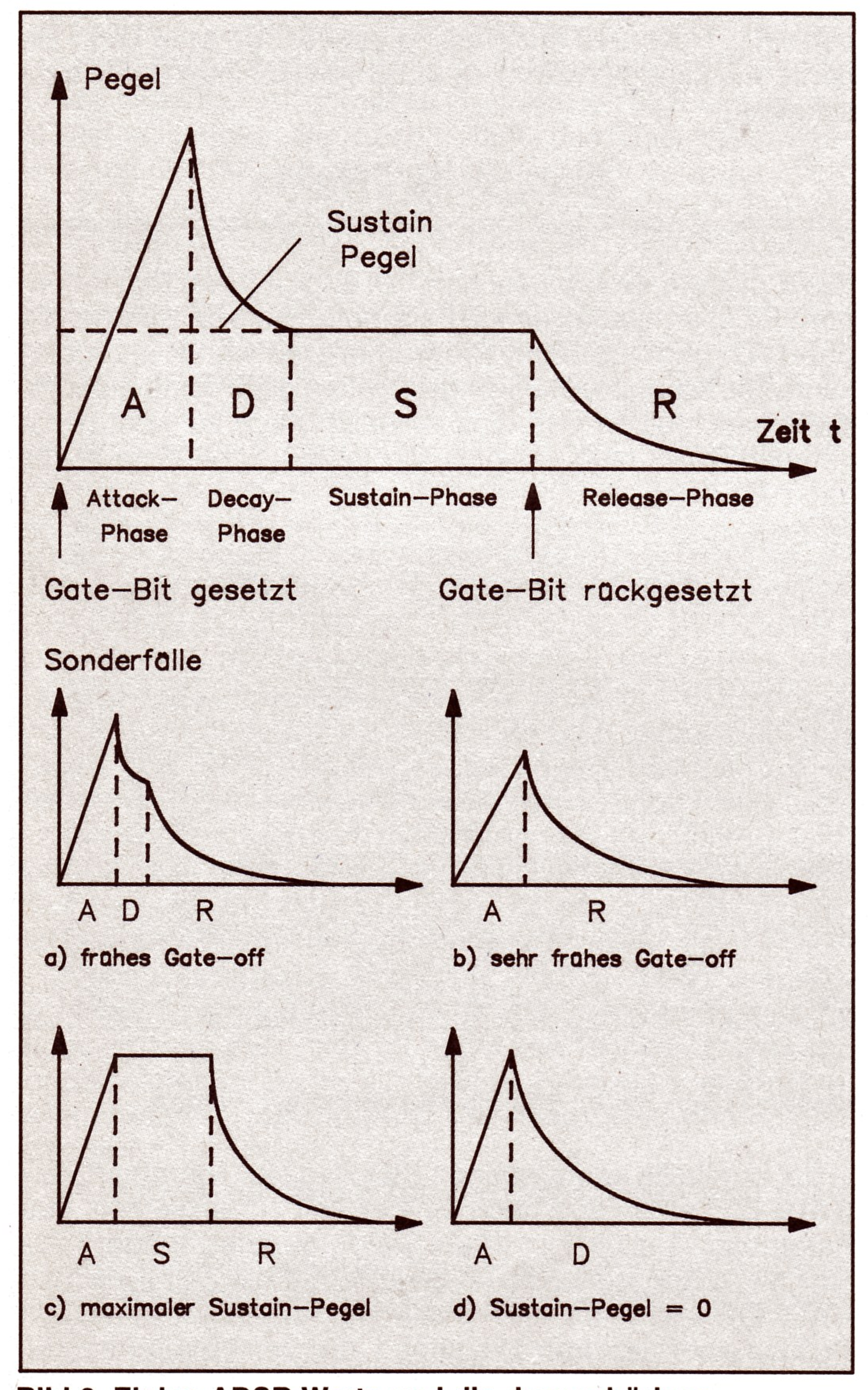



**Bild 3. Einige ADSR-Werte und die dazugehörigen Hüllkurven**

Und dieses Obertonspektrum kann man mit dem Filter verändern. Der Filter im SID kennt dazu drei Betriebsarten, die über die Bits 4, 5 und 6 in Register S+24 gewählt werden:

**Tiefpaß** Frequenzen (Obertöne) oberhalb der in den Registern S+21 und S+22 einstellbaren Filterfrequenz werden abgeschwächt, und zwar um so mehr, je höher diese Frequenzen sind. Der Gesamtklang wird dadurch dunkler und weicher.

**Hochpaß** Es werden die Frequenzen abgeschwächt, die unterhalb der Filterfrequenz liegen. Höhere Frequenzen werden ungehindert durchgelassen. Mit einem Hochpaß kann man den Grundton eines Klanges abschwächen. Seine Gesamtzusammensetzung verschiebt sich dann zugunsten der Obertöne. Der Klang wird dabei dünner und heller.

**Bandpaß** Dieser Filtermodus schwächt Frequenzen auf beiden Seiten der Filterfrequenz ab. Der Klang wird dabei, wie man fast erwarten kann, etwas dürftig, sofern man nicht maximale Resonanz (siehe weiter unten) einstellt.

## Filterfrequenz (Register S+22 und S+23)

Die Filterfrequenz kann auf elf Bit genau eingestellt werden. Dabei kann man aber die drei niederwertigen Bits in Register S+21 unberücksichtigt lassen, da ihr Einfluß praktisch unhörbar ist.

## Resonanz und Stimmen-Wahlschalter (Register S+23)

Über den 4-Bit-Parameter Resonanz kann ein gefilterter Klang effektvoller gestaltet werden. Bei großer Resonanz (der Maximalwert ist 15) werden Frequenzanteile in der Gegend der Filterfrequenz verstärkt. Die sonstigen abschwächenden Eigenschaften von Tief- und Hoch- und Bandpaß bleiben dabei erhalten.

Über die Bits 0, 1 und 2 desselben Registers kann man für jede der drei SID-Stimmen unabhängig wählen, ob sie gefiltert oder ungefiltert erklingen soll. Ist zum Beispiel Bit 0 gesetzt, so wird Stimme 1 gefiltert. Das Bit 3, Filter Ex, steuert die Verarbeitung einer von außen zuführbaren Signalquelle, zum Beispiel eines zweiten SID.

## Filtermodus und Lautstärke (Register S+24)

Die Lautstärkeneinstellung haben wir schon kennengelernt. Man wird sie meistens auf ihren Maximalwert 15 stellen, weil dann der Rauschabstand und damit die Klangqualität am besten ist. Durch Setzen der Bits 4, 5 und 6 wird die Betriebsart des Filters, Tief-, Band- oder Hochpaß gewählt. Die Betriebsarten sind uneingeschränkt kombinierbar. Mit Bit 7 (S3 Aus) kann man die Stimme 3 unhörbar machen. Der Sinn dieser Funktion wird in folgendem Abschnitt klar.

Ein Beispiel zur Filterprogrammierung:
Stimme 1 soll mit maximaler Resonanz durch den Tiefpaßfilter geschickt werden:

```
FF  = 50       Filterfrequenz (nur High-Byte)
FR  = 241      (=15*16 für Resonanz +1 für Bit 1)
ML  = 31       (=16 für Bit 4 + 15 für Lautstärke)
POKE S+22,FF
POKE S+23,FR
POKE S+24,ML
```

## Die Leseregister S+25 bis S+28

Die Register S+25 und S+26 haben mit der Klangprogrammierung nichts zu tun. Über sie können die Werte zweier an Joystick-Ports angeschlossener Potentiometer (Paddles)

| Register | A S+5 | D S+5 | SU S+6 | R S+6 | C S+4 | P S+2 S+3 | F S S+1 | FF S+22 | FR S+23 | ML S+24 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Glöckchen | 0 | 10 | 0 | 10 | 16 | x | 40000 | x | 0 | 15 | 100 |
| Oboe | 8 | 7 | 10 | 8 | 64 | 250 | 7500 | x | 0 | 15 | 500 |
| Fagott | 8 | 7 | 10 | 8 | 64 | 250 | 2500 | x | 0 | 15 | 750 |
| Zungenpfeife | 8 | 0 | 15 | 10 | 48 | x | 400 | x | 0 | 15 | 1000 |
| Banjo | 0 | 8 | 0 | 8 | 32 | x | 7500 | 50 | 241 | 111 | 30 |
| Stahl | 0 | 0 | 15 | 12 | 96 | 2044 | 30000 | x | 0 | 15 | 100 |
| Feder | 0 | 8 | 0 | 9 | 32 | x | 750 | x | 0 | 15 | 35 |
| Preßlufthammer | 0 | 0 | 15 | 10 | 80 | 2100 | 200 | x | 0 | 15 | 2000 |
| Schuß | 0 | 8 | 0 | 10 | 128 | x | 10000 | x | 0 | 15 | 50 |
| Starkstrom | 0 | 0 | 15 | 0 | 128 | x | 100 | x | 0 | 15 | 2000 |
| Düsenflugzeug | 0 | 0 | 15 | 13 | 128 | x | 3000 | 50 | 241 | 31 | 3000 |
| Rakete | 0 | 0 | 15 | 15 | 128 | x | 1000 | 10 | 241 | 31 | 3000 |

x = don't care (Parameter muß nicht eingestellt werden)

**Tabelle 1. Einige Beispiel-Effekte für Listing 1. Die Parameter müssen in den Zeilen 320 bis 390 verändert werden.**

abgefragt werden. Interessant sind die Register S+27 und S+28:

Aus S+27 kann man den Signalverlauf von Stimme 3 in Form von Byte-Werten lesen. Mit folgendem kleinen Programm kann man diesen Signalverlauf sogar sichtbar machen:

```
10 S=54272
20 POKE S+14,10        :REM F LOW
30 POKE S+15,0         :REM F HIGH
40 POKE S+18,16        :REM DREIECK
50 PRINT TAB(PEEK(S+27)/7);" *":GOTO 50
```

Hier sollte man einmal ein wenig mit den Parametern in Zeile 20–40 experimentieren.

Auf die gleiche Weise kann man aus Register S+28 den Hüllkurvenverlauf von Stimme 3 lesen.

```
100 S=54272
110 POKE S+19,16+11+11     :REM A D
120 POKE S+20,16*8 +11     :REM S R
130 POKE S+18,1            :REM GATE ON
140 FOR I=1 TO 50
150 PRINT TAB(PEEK(S+28)/7);" *"
160 NEXT I
170 POKE S+18,0            :REM    GATE OFF
180 FOR I=1 TO 50
190 PRINT TAB(PEEK(S+28)/7);" *"
200 NEXT I
```

Diese Werteverläufe sind besonders zum Modulieren anderer Stimmen geeignet. Normalerweise möchte man Stimme 3 dann nicht hören, wenn man ihren Signalverlauf oder ihre Hüllkurve anderweitig verwendet. Man kann sie dann, wie schon erwähnt, über Bit 7 in Register S+24 ausschalten.

## Klangvolle Effekte

Nach diesem systematischen Teil folgen noch Einstellungen. Tabelle 1 enthält einige Parametersätze für Klänge, die der SID ohne großen Programmieraufwand erzeugen kann. Die Klangbezeichnungen wollen die Effekte nur subjektiv beschreiben und sind natürlich nicht zu wörtlich zu nehmen. Man muß lediglich die Werte einer Zeile in die in der Kopfzeile angegebenen SID-Register schreiben, das GATE-Bit setzen und nach einiger Zeit zurücksetzen. Der Parameter M ist übrigens kein SID-Parameter, sondern soll eine Verzögerungsschleife steuern, die die Zeit zwischen GATE ON und GATE OFF bestimmt. Parameter M bezieht sich auf das Programm in Listing 1, das beim Experimentieren Hilfestellung leisten soll. Im DATA-Teil ab Zeile 320 sind die Parameter aus Tabelle 1 einzusetzen, und zwar genau in der gleichen Reihenfolge. Das Programm belegt nach dem Start den SID mit den Parametern aus den DATA-Zeilen und wartet auf einen beliebigen Tastendruck, der dann den Klangeffekt auslöst. Man versuche es auch einmal mit den Tasten, die eine Auto-Repeat-Funktion haben, wie zum Beispiel die Space-Taste.

Das Programm aus Listing 2 ist ganz ähnlich aufgebaut, kann aber ein viel größeres Spektrum von Effekten dadurch realisieren, daß es die Frequenz von Stimme 1 dynamisch

| Fest eingestellt: Register | FF S+22 | FR S+23 | ML S+24 |
|---|---|---|---|
| | x | 0 | 15 |

| Register | A S+5 | D S+5 | SU S+6 | R S+6 | C S+4 | P S+2 S+3 | F S S+1 | G | N | M |
|---|---|---|---|---|---|---|---|---|---|---|
| Telefon | 0 | 10 | 0 | 10 | 16 | x | 16000 | 1.33 | 2 | 25 |
| Laserkanone | 0 | 0 | 15 | 0 | 64 | 1000 | 30000 | 0.85 | 10 | 1 |
| Take-Off | 0 | 0 | 15 | 15 | 128 | x | 500 | 1.004 | 1000 | 1 |
| Turbine | 0 | 0 | 15 | 15 | 96 | 2044 | 20000 | 1.001 | 460 | 1 |
| Trommelwirbel | 0 | 5 | 2 | 9 | 128 | x | 20000 | 1 | 2 | 30 |
| Maschinengewehr | 0 | 5 | 2 | 9 | 128 | x | 12000 | 0.7 | 3 | 30 |

x = don't care (Parameter muß nicht eingestellt werden)

**Tabelle 2. Einige Beispiel-Effekte für Listing 2. Die Parameter müssen in den Zeilen 420 bis 480 verändert werden.**

| Register | A | D | SU | R | C | P | F | G | N | M | A3 S+19 | B3 S+19 | F3 S+20 | R3 S+20 | C3 S+18 | P3 S+16 S+17 | F3 S+14 S+15 | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vogelgezwitscher | 0 | 8 | 0 | 8 | 16 | x | 40000 | 500 | 8 | 10 | 0 | 8 | 0 | 0 | x | x | x | 28 |
| Bongo | 0 | 7 | 0 | 7 | 16 | x | 4000 | 1000 | 4 | 1 | 0 | 8 | 0 | 0 | x | x | x | 28 |
| E-Baß | 0 | 8 | 0 | 9 | 32 | x | 750 | 1000 | 7 | 1 | 0 | 10 | 0 | 0 | x | x | x | 28 |
| Dampfhammer | 0 | 9 | 0 | 11 | 128 | x | 5000 | 200 | 20 | 15 | 0 | 9 | 0 | 9 | x | x | x | 28 |
| Martinshorn | 0 | 0 | 15 | 8 | 64 | 1000 | 7000 | 720 | 300 | 1 | x | x | x | x | 64 | 2048 | 15 | 27 |
| Sirene | 10 | 13 | 0 | 0 | 64 | 2048 | 10000 | 400 | 300 | 1 | x | x | x | x | 16 | x | 30 | 27 |
| Geklimper | 0 | 0 | 15 | 0 | 64 | 2048 | 10000 | 200 | 400 | 1 | x | x | x | x | 128 | x | 20 | 27 |
| Grollen | 9 | 10 | 0 | 0 | 32 | x | 50000 | 2000 | 40 | 10 | x | x | x | x | 128 | x | 500 | 27 |

**Tabelle 3. Einige Beispiel-Effekte für Listing 3. Die Parameter müssen in den Zeilen 610 bis 650 verändert werden.**

verändert. Dazu dient die innere Schleife, Zeile 260–300. Dort wird bei jedem Durchlauf die Frequenz F1 mit einem Faktor G multipliziert. Für G > 1 steigt die Frequenz schneller an, für G < 1 nimmt sie ab und zwar um so schneller, je weiter G von 1 entfernt ist. Die Umrechnung von F1 in Low- und High-Byte geschieht hier nach der schnellen Methode 2. Man beachte, daß damit nur F1-Werte bis 32767 verarbeitet werden können, also nur die Hälfte des vollen Frequenzumfangs des SID. In der äußeren Schleife wird der Wert von F1 auf seinen Ausgangswert F zurückgesetzt. Außerdem steuert die äußere Schleife bei jedem Durchlauf einen ADSR-Hüllkurvenzyklus durch GATE-ON-GATE-OFF. Die Zahl N gibt dabei die Anzahl der inneren Schleifendurchläufe an, die Zahl M die der äußeren Schleifendurchläufe. Beispielparameter zu Listing 2 findet man in Tabelle 2.

Das dritte Programm (Listing 3) verwendet schließlich den Signalverlauf oder die Hüllkurve von Stimme 3, um Stimme 1 zu modulieren. Dies geschieht in der inneren Schleife, Zeile 360–380. Q ist dabei die Adresse eines der Leseregister des SID, also entweder S+27 für den Signalverlauf oder S+28 für den Hüllkurvenverlauf. In der DATA-Zeile 650 ist dazu nur 27 oder 28 anzugeben. Über die Variable G kann die Stärke der Modulation, die sogenannte Modulationstiefe, gesteuert werden. Ein kleinerer Wert von G ergibt hier eine stärkere Modulation. Die äußere Schleife steuert hier ADSR-Hüllkurvenzyklen für Stimme 1 und Stimme 3. Dieses kleine Programm sollte Anlaß zum weiteren Experimentieren sein. Man kann statt der Tonfrequenz zum Beispiel auch einmal versuchen, die Pulsweite oder die Filterfrequenz zu modulieren. Tabelle 3 enthält einige Parametersätze für dieses Programm.

Dieses Thema wollen wir hier nur einmal streifen. Grundlage hierfür ist die genaue Kenntnis der Tonleiterfrequenzen. Diese müssen aber nicht mühsam aus Tabellen abgetippt, sondern können durch ein kleines Programm selbst berechnet werden. Man muß dazu ein klein wenig Mathematik betreiben:

1) Die Frequenzen zweier Töne im Oktavabstand verhalten sich wie 2:1.
2) Eine Oktave ist durch die Halbtöne in zwölf gleiche Intervalle eingeteilt und zwar nicht linear, sondern exponentiell.
3) Das Frequenzverhältnis H zweier aufeinanderfolgender Halbtöne ist die zwölfte Wurzel aus zwei. In Basic läßt sich das leicht ausrechnen:

   $H = 2^{1/12}$
4) Man bekommt dann alle Halbtöne, indem man die Frequenzen, ausgehend von einem Grundwert, fortlaufend mit H multipliziert.

Im Programm aus Listing 4 macht das eine Schleife in Zeile 140 bis 190. Die Variable FAUS wird mit 110 vorbesetzt. Das entspricht einem »großen A«, dem Ton, der zwei Oktaven unter dem Kammerton, dem bekannten »eingestrichenen a« mit 440 Hz, liegt. In Zeile 170 wird FAUS in den dazugehörenden SID-Wert F umgerechnet. F wird in High- und Low-Byte zerlegt, welche in den Feldern FH und FL gespeichert werden. Dort stehen die Töne als fertige POKE-Werte zum Spielen einer Melodie zur Verfügung.

Das restliche Programm erzeugt aus diesen Werten eine mehr oder weniger zufällige Tonfolge auf dem SID. Ein wenig wird der Zufall aber durch die Werte in den DATA-Zeilen ab Zeile 500 gesteuert. Diese Werte stellen ein Blues-Schema in codierter Form dar. Das Schema selbst steht in den letzten drei Zeilen. Das Programm holt sich aus diesem Schema die erste Zahl. Diese zeigt auf eine der sieben Auswahlmengen in den vorausgehenden Zeilen. Die Auswahlmengen enthalten Tonnummern. Das Programm spielt nun nacheinander acht zufällig ausgewählte Töne aus dieser Menge, Tonwiederholungen sind dabei möglich. Anschließend geht das Programm über den nächsten Zeiger aus dem Schema zu einer neuen Auswahlmenge über. Die links etwas abgesetzten Zahlen dienen dabei nur zur Längenangabe der Auswahlmengen und des Schemas.

Mit dieser einfachen Technik wird der Blues-Charakter deutlich hörbar.

## Computer und Musik

In jüngster Zeit hat die Musik mehr und mehr Einzug in die Welt der Elektronik gehalten. Professionelle Musiker und Arrangeure wie Giorgio Moroder und Jean-Michelle Jarre komponieren ihre erfolgreichen Hits zum großen Teil auf entsprechend ausgerüsteten Computern. Komponisten und Informatiker wirken gleichermaßen gestaltend: Sie schreiben ihre Werke in einer abstrakten Sprache. Erst mit der Interpretation durch Instrumente oder einen Computerbaustein (Soundchip) erwachen diese Werke zum Leben und lassen ihre Faszination auf uns wirken. Sowohl in der Informatik als auch in der Musik gab und gibt es Tendenzen, universelle Instrumente zu schaffen, die möglichst alles können. Die gewaltigen Kirchenorgeln und Orchestrions des vorigen Jahrhunderts zeugen in der Musikgeschichte von diesen Bemühungen. Programmgesteuerte Computer waren bald nach ihrer Erfindung mehr als stupide Rechenmaschinen. Heutzutage ist die technische Annäherung der beiden Bereiche so weit fortgeschritten, daß es keinen wesentlichen Unterschied zwischen Computern und Musikinstrumenten gibt. Mit dem Computer lassen sich Noten editieren oder Partituren auf einem Plotter ausgeben. Ein moderner Synthesizer kann nicht nur typisch »elektronische« Klänge, sondern Instrumente und menschliche Stimmen nachahmen.

Zumindest der Studiomusiker kann heute nicht mehr auf die Computertechnik verzichten. Der Computerspezialist beschäftigt sich eher spielerisch mit den Soundmöglichkeiten.

Viel Spaß beim Programmieren eigener »Sphärenklänge« und Pop-Hits!

(Thomas Krätzig/bl)




**Eröffnen Sie Ihr eigenes Tonstudio. Nutzen Sie den fantastischen Synthesizer-Chip in Ihrem C64: von der live eingespielten Melodie bis zur professionellen Komposition.**

Der Soundchip des C64 bietet beachtliche Möglichkeiten, leider läßt er sich ohne Tools zu schwer programmieren. Um seine immensen Fähigkeiten ausschöpfen zu können, wurde ein »Monitor« entwickelt, der speziell die Eingabe von Musikdaten unterstützt: der »Soundmonitor« (Bild 1). Mit den entsprechenden Daten kommen Ergebnisse zustande, die sogar Musikuntermalungen aus professionellen Spielen übertreffen.

Der »Soundmonitor« unterscheidet sich in einigen wesentlichen Merkmalen von anderen Soundeditoren. Sein Hauptteil, die Abspielroutine, läuft völlig selbständig im Interrupt. Das bedeutet, zu jeder Zeit kann der Song angehört werden. Außerdem enthält er ein »Realtimerecord« (Aufnahme von Musik während des Spielens auf der Tastatur).

Laden Sie das Programm von der beiliegenden Diskette mit:

```
LOAD"SOUND-MONITOR",8
```

und starten Sie mit RUN.

Anschließend befinden Sie sich im Hauptmenü des »Sound-Monitors«, im folgenden »Track/Step-Table« genannt (Bild 1).

Links oben sehen Sie unter dem »T« ein blinkendes Quadrat, den Cursor. Erreicht der Cursor den oberen Rand, so scrollt das Zahlenfeld nach unten (am unteren Rand nach oben). Diese Zahlendarstellungen erscheinen wie alle folgenden in hexadezimaler Schreibweise. Das bedeutet, eine Zahlenreihe beinhaltet nicht wie in dezimaler Schreibweise zehn Werte (0 bis 9), sondern 16 Zahlenwerte (0 bis 9 und A bis F).

**<SHIFT I> - Speicherbereich vorbereiten (INITIALIZE)**

Die wichtigste Vorbereitung ist das »Initialisieren« des Musikdatenspeichers. Der Aufruf dieser Funktion erfolgt durch Drücken der Tastenkombination <SHIFT I> und Bestätigen der Sicherheitsrückfrage mit <Y>. Danach erscheint wieder das Anfangsbild.

**<X> - Programm verlassen (EXIT)**

Die Taste <X> beendet das Programm. Mit »SYS 4096« wird der Soundmonitor ohne Datenverlust wieder gestartet.

**<Z> - Soundparameter**

Durch Drücken der Taste <Z> erreichen Sie die Soundpage (Soundseite). Bevor Sie Ihrem C64 einen Ton entlocken können, müssen Sie zumindest eine Soundpage editiert haben. Beachten Sie bitte rechts oben die »Sound Number«. Die Tasten <F1> und <F3> erlauben es Ihnen, 256 unterschiedliche Sounds zu kreieren ($00-$FF). Die Sound Number steht beim ersten Aufruf auf »00«. Dieser Sound ist nach dem Initialisieren als Metronom belegt (siehe »Metronom«).

Pro Sound lassen sich 24 Register editieren:

0. waveform (keyon)

Die Wellenform des Tons. Mögliche Werte sind 11, 21, 41 und 81.

Die 1. Ziffer: 0 = kein Klang, 1 = Dreieck, 2 = Sägezahn, 4 = Rechteck, 8 = Rauschen

Die 2. Ziffer: funktioniert bitweise:
Bit 0 = Anschlagbit (muß 1 sein, startet Hüllkurve)
Bit 1 = Synchronbit
Bit 2 = Ringmodulation1
Bit 3 = Testbit

1. attack/decay

Anstieg/Abstieg des Anschlags. Entspricht Register 5, 12 und 19 des SID (Zeichen siehe Tabelle rechts ▶).

2. sustain/release

Relative Lautstärke /Ausklingzeit des Tons. Entspricht Register 6, 13 und 20 des SID (Bild 1).

Die 1. Ziffer (sustain): 0 = unhörbar, F = volle Lautstärke
Die 2. Ziffer (release): entspricht der Wertereihe von »decay«.

3. portamento effectbyte

Effekt für Puls EG mode (siehe Register 9)




**Bild 1. Das »Gesicht« des Soundmonitors**

4. pulserate

Pulsrate bezeichnet das Tastverhältnis der Rechteckwelle. Werte zwischen 00 und FF sind erlaubt.

5. pulse EG count up time

Bei der Modulation des Tastverhältnisses entsteht ein schwebender Klang. Dieses Register gibt an, wie lange der Wert aus Register 7 zum Tastverhältnis addiert wird.

6. pulse EG count down time

Bei der Modulation des Tastverhältnisses entsteht ein schwebender Klang. Mit diesem Register wird angegeben, wie lange der Wert aus Register 7 vom Tastverhältnis subtrahiert wird.

| 1. Ziffer (attack): | 2. Ziffer (decay): |
|---|---|
| 0 = 0.002 s | 0 = 0.006 s |
| 1 = 0.008 s | 1 = 0.024 s |
| 2 = 0.016 s | 2 = 0.048 s |
| 3 = 0.024 s | 3 = 0.072 s |
| 4 = 0.038 s | 4 = 0.114 s |
| 5 = 0.056 s | 5 = 0.168 s |
| 6 = 0.068 s | 6 = 0.204 s |
| 7 = 0.080 s | 7 = 0.240 s |
| 8 = 0.100 s | 8 = 0.300 s |
| 9 = 0.250 s | 9 = 0.750 s |
| A = 0.500 s | A = 1.500 s |
| B = 0.800 s | B = 2.400 s |
| C = 1 s | C = 3 s |
| D = 3 s | D = 9 s |
| E = 5 s | E = 15 s |
| F = 8 s | F = 24 s |

7. pulse EG count level
Die Höhe des Wertes wird zu dem Register 5 addiert bzw. Register 6 subtrahiert. Ein Wert über 30 läßt den Ton unsauber erklingen.
8. waveform (keyoff)
Entspricht Register 0. Das Anschlagbit (Bit 0, rechte Ziffer) ist grundsätzlich auf 0 zu setzen. Die linke Ziffer beschreibt die Wellenform (siehe Register 0). Mögliche Werte: 10, 20, 40 und 80. Mit diesem Register ist es möglich, die Wellenform während eines Tones zu wechseln. Im Normalfall ist die Wellenform die gleiche wie in Register 0.
9. puls EG mode/portamento effect
Ziffer 1 ist der Neustart von Puls EG. Es sind nur zwei Werte möglich: 0 = ausgeschaltet, 1 = eingeschaltet.
Ziffer 2 wird bitweise bearbeitet:
Bit 0 = Ton nach oben ziehen
Bit 1 = Ton nach unten ziehen
Bit 2 = Ton wird nur einmal (=1) oder ständig (=0) gezogen.
Bit 3 ist ohne Funktion.
Die Grenzen, zwischen denen der Ton gezogen wird, sind die angeschlagene Note und das Register 3.
10. portamento level low
Portamento ist ein Ziehen der Tonhöhe von einem Ton zum anderen. In diesem Byte wird die Geschwindigkeit (low) bestimmt. 00 ist kein Portamento.
Vergessen Sie nicht für diesen Effekt das Portamentobyte zu setzen (siehe »SOUND-OPTIONS«).
11. portamento level high
Geschwindigkeit (high) des Portamentos. Dieser Wert führt, wenn er hoch genug ist und Bit 0 von Register 9 gesetzt ist, zu interessanten Effekten.
12. vibrato level
Intensität des Vibratos
13. vibrato speed
Geschwindigkeit des Vibratos. 00 = schnell, 7F = langsam. Bit 7 entscheidet über die Richtung. 0 = nach oben, 1 = nach unten.
14. vibrato deley
Zeitliche Verzögerung vom Anschlag des Tons bis zum Beginn des Vibratos.
15. fine detune
Feineinstellung der Tonhöhe (verstimmen). Normaleinstellung ist 00.
16. filter (high nibble of sid reg24)
Wird kein Filter benötigt, erhält dieses Byte den Wert FF. In diesem Fall werden andere Stimmen, die Filter benötigen, nicht beeinflußt. Steht dieses Register auf »00«, werden bei Anschlag dieser Stimme die Filter ausgeschaltet und es kann zu einem Knacken des »SID« kommen.
Ansonsten ist nur die linke Ziffer von Bedeutung (Ziffer 1):
Bit 0 = Tiefpaßfilter (1 = ein)
Bit 1 = Bandpaßfilter (1 = ein)
Bit 2 = Hochpaßfilter (1 = ein)
17. resonance/filter to voice (reg23)
Entspricht Reg. 16 des SID.
Die 1. Ziffer ist die Filterresonanz.
Die 2. Ziffer schaltet die zu filternden Stimmen ein (Bit ist gesetzt).

## Kurzinfo: Soundmonitor

**Programmart:** Musik-Editor
**Laden:** LOAD"SOUND-MONITOR",8
**Starten:** nach dem Laden RUN eingeben
**Steuerung:** über Tastatur
**Besonderheiten:** erzeugt eigenständiges Musikabspielprogramm
**Benötigte Blocks:** 51 Blocks, 4 Blocks Hilfsprogramme und 45 Blocks pro Komposition
**Programmautor:** Chris Hülsbeck

Bit 0 = Stimme 1
Bit 1 = Stimme 2
Bit 2 = Stimme 3
18. filter cut off frequency
Filterfrequenz zwischen »00« und »FF«.
19. filter EG count up time
Gibt die Zeiteinheit an, bei der die Filterfrequenz zu Register 22 addiert wird.
20. filter EG count down time
Siehe Register 19, nur wird hier von Register 22 subtrahiert.
21. f. EG count level low (up/down)
Die 1. Ziffer (0-F) bestimmt den Wert, um den nach oben gezählt wird. Die 2. Ziffer ist der Betrag, um den nach unten gezählt wird. Nur für Effekte mit Register 23.
22. f. EG count level high
Der Wert, um den nach unten bzw. nach oben gezählt wird. »00« bis »20« sind sinnvoll.
23. f. EG mode/trigge voice
Die 1. Ziffer:
Bit 0 = Filter EG wird bei jedem Anschlag neu gestartet (=1)
Bit 1 = Anfang bei »count up« oder »count down«.
Bit 2 bis 3 sind unbelegt.
Die 2. Ziffer:
Entscheidet über die Stimme, die starten soll.
Bit 0 = Stimme 1
Bit 1 = Stimme 2
Bit 2 = Stimme 3
Bit 3 ist unbelegt.
Sie verlassen die Soundpage durch Drücken von <RETURN>.

**Der TRACK/STEP-TABLE**

In der zweiten Bildschirmzeile des Titelbilds stehen Abkürzungen für die darunterliegenden Zahlenkolonnen:
SP (STEP).
Ein Step ist ein gespielter Takt. Unter »SP« ist die Nummer des Steps angezeigt. Die Wertereihe ist fest vorgegeben und reicht von $00 bis $FF (256 Werte).

**TRK1 - TRK3 (Track 1)**

»TRK1« heißt Track1 und ist nichts weiter als die Abspielfolge der Takte für die Stimme 1 des Soundchips. »TRK2« und »TRK3« sind für Stimme 2 bzw. 3 zuständig. Diese Tracks ermöglichen es, beliebige Tonfolgen live einzuspielen oder zu editieren. Sie erreichen diese Tracks mit dem Cursor. Durch Überschreiben ist es möglich, andere Speicherbereiche zu definieren. Nach einem Initialize sind alle Steps mit dem Takt $BE00 belegt. Dieser enthält keine Noten und dient als Leertakt (z.B. für Pausen). Der Bereich für die Takteingaben liegt zwischen $B000 und $BE00. Hier geht Ihnen kein Basic-Speicher verloren. Um Ihnen eine Übersicht der verwendbaren Bereiche zu geben, zeigt Tabelle 1 die Speicheraufteilung des Soundmonitors.
Stellen Sie den Cursor auf die Adresse des Taktes, den Sie editieren wollen. Um dann in die Notendarstellung des Taktes zu gelangen, drücken Sie <RETURN>. Auf dem Bildschirm erscheint jetzt der Takt.

```
BE00   ---  00   ---  00   ---  00   ---  00
BE08   ---  00   ---  00   ---  00   ---  00
BE10   ---  00   ---  00   ---  00   ---  00
usw.
```

Dieser Takt wird von links oben nach rechts unten abgearbeitet. Die oben dargestellten drei Bindestriche können (mit der Tastatur) durch Notenbezeichnungen überschrieben werden. Es sind jeweils die Noten (C bis H), die Halbnotenverschiebung (»#«) sowie die Oktave (0 bis 7) erlaubt. Bleiben die Striche erhalten, ergibt sich für diesen Takt eine Pause. - Bei der Eingabe der Noten (ohne <SHIFT>) erscheinen

die Notenbezeichnungen invers. Damit wird gekennzeichnet, daß dieser Ton beim Abspielen angeschlagen wird. Soll dies nicht geschehen (z.B. beim »Ziehen« der Tonhöhe), erfolgt die Eingabe zusammen mit <SHIFT>.

- Drei Striche bedeuten Pause. Eingegeben wird durch <-> auf der ersten Stelle einer Note.
- Drei Pluszeichen bedeuten, daß der Ton gehalten wird.

Zu jeder Note gehört noch eine zweistellige Hexadezimalzahl (»00«), deren unteres Nibble (rechte Ziffer) die Soundnummer angibt (siehe <Z>). Auf diese Art sind die ersten 16 Sounds (0 bis F) der voreingestellten Klänge erreichbar.

Die Erklärung des oberen Nibbles finden Sie unter »SOUND-OPTIONS«.

Verlassen wird diese Page, indem Sie <RETURN> drükken.

**TR (Transpose)**

Transpose bedeutet das Wandeln in eine andere Tonhöhe. Das Transponieren geschieht in dem Track neben »TR«. Damit wird es möglich, jede Stimme einzeln in der Tonhöhe zu variieren. Zu den einzelnen Noten wird der angegebene Wert addiert (00 bis 7F = positives Transpose, FF bis 80 = negatives Transpose).

Aus einem C-2 wird bei einem Transpose von:

01 ein C#4
03 ein D#4
0C ein C-3
FF ein H-1
FB ein G-1

**ST (Soundtranspose)**

Das Soundtranspose-Byte verändert im Gegensatz zum Transpose-Byte nicht die Stimmenhöhe, sondern addiert sich zu jeder Soundnummer im Takt. Dadurch wird es möglich, mehr als 16 Sounds anzuwählen oder einen einzelnen Takt mit verschiedenen Klängen zu spielen.

**AR/S (Grundeinstellungen und Arpeggios)**

In der AR/S-Page werden Tempo, Taktlänge sowie Lautstärke eingestellt. Die Speicheradresse kann beliebig verändert werden. In die Darstellung gelangen Sie mit <RETURN>. Dabei muß der Cursor in der AR/S-Spur stehen. Es wird der Speicherbereich unter dem Cursor genommen. Die 8 Byte der ersten Zeile sind für die Grundparameter zuständig und haben folgende Bedeutung:

- Byte 0: Geschwindigkeit des CIA-Timers low (Feineinstellung des Tempos, z.B. zum Synchronisieren mit einem Plattenspieler, siehe »SYCHRONISATION«)
- Byte 1: Geschwindigkeit des CIA-Timers high (möglichst 35 bis 50)
- Byte 2: Grobeinstellung des Tempos (zwischen 0 und 4) (für alle Geschwindigkeitseinstellungen gilt: je kleiner die Zahl, desto höher das Tempo)
- Byte 3: Taktlänge

Die Taktlänge ist nach einem Initialize auf den Wert $20 eingestellt (dezimal 32). Das bedeutet, daß der Takt 32 Noten enthalten kann und entspricht einem 4/4-Takt. Da zu jeder Note aber noch eine Soundnummer dazugehört, ist der Datenblock doppelt so lang (beginnt beispielsweise der erste Takt bei $B000, dann startet der zweite Takt bei $B040).

Andere Taktlängen:

$30 = 6/8-Takt: erster Takt $B000, zweiter Takt $B030.
$40 = wie $20 ein 4/4-Takt, aber doppelte Länge: $B000, $B080

- Byte 4: Zuständig für ein langsames Ausklingen am Ende des Musikstücks (FF = kein Ausklingen). Um diese Funktion nutzen zu können, ist am Ende des Liedes bei »AR/S« eine Adresse anzugeben, die die gleichen Grundeinstellungen und Arpeggios hat. Lediglich das 4. Byte darf sich vom alten AR/S unterscheiden (Werte von 10 bis 30 sind empfehlenswert, 10 = schnelles Ausklingen, 30 = langsames Ausklingen)
- Byte 5: Lautstärke 00 bis 0F (identisch mit SID-Register 24).
- Byte 6 und Byte 7 sind unbenutzt.

**Editor-Befehle des Track/Step-Table**

**<SHIFT INS/DEL> Insert Step (Step einfügen)**

Fahren Sie mit den Cursortasten auf den Step im Track/Step-Table, vor dem Sie einfügen wollen. Anschließend betätigen Sie <SHIFT INS/DEL>.

**<INST/DEL> Delete Step (Step entfernen)**

Bewegen Sie den Cursor über den zu entfernenden Step und drücken Sie <INST/DEL>.

**<SHIFT @> Clear Step (Step löschen)**

Zum Löschen bewegen Sie den Cursor über den entsprechenden Step und drücken <SHIFT @>.

**Copy**

Der Soundmonitor kennt drei Arten von Kopierfunktionen:

1. <SHIFT => - Kopieren mehrerer Tracks

Um einen oder mehrere Steps im Track/Step-Table an eine andere Stelle zu kopieren, müssen folgende Bedingungen erfüllt sein:

- »FIRST STEP« stellt den ersten zu kopierenden Step dar.
- »LAST STEP« den letzten.
- Der Cursor muß auf dem Step stehen, an den der Block kopiert werden soll.

Mit <SHIFT => kopieren Sie nun die markierten Blocks in den Bereich ab Cursor.

2. <SHIFT L> / <SHIFT S> - Kopieren von Soundparametern

Sie kopieren eine Soundnummer, indem Sie die Sounddarstellung anwählen (<Z>).

- Mit <F1> und <F3> wählen Sie den zu kopierenden Sound an.
- <L> lädt den Sound in den Copy-Speicher.
- Mit <F1> und <F3> suchen Sie die Soundnummer aus, in die kopiert werden soll.
- <SHIFT S> überträgt die Daten in diesen Bereich.

3. <SHIFT L> / <SHIFT S> - Kopieren eines Taktes

Zum Kopieren eines Taktes wählen Sie die Notendarstellung an (mit dem Cursor über den zu kopierenden Speicherbereich fahren und <RETURN> drücken).

- <L> färbt die Rahmenfarbe rot und lädt den Takt in den Copy-Speicher.
- Verlassen Sie die Notendarstellung mit <RETURN>
- Wählen Sie den Takt an, in den Sie kopieren wollen.
- Drücken Sie <SHIFT S>. Der alte Takt wird in den neuen übernommen.

In der Arpeggiodarstellung kann mit den gleichen Tasten jeweils eine Zeile kopiert werden.

**Funktionen des Track/Step-Tables**

**Arpeggios**

Arpeggios sind schnelle Tonhöhenänderungen, mit denen man Akkorde simulieren kann. Sie werden in der Zeile unter den Grundeinstellungen gesetzt. Ein »Arpeggio« besteht immer aus acht Notensteps, die vom Programm nacheinander aufgerufen werden. Der Grundton wird in einem Takt angeschlagen. Die einzelnen Werte werden wie bei »Transpose« zur Grundnote addiert. Ein Moll-Akkord sieht folgendermaßen aus:

0c 07 03 00 0c 07 03 00

So kann fast jeder Akkord simuliert werden. Um dieses Arpeggio aufzurufen, muß in der Taktpage das Bit 3 der ersten Ziffer hinter der Note gesetzt sein (Wert = 4).

Für jede Note ist im Takt der Speicherplatz des Arpeggio festzulegen. Da es sehr unwahrscheinlich ist, daß zwei Noten direkt hintereinander gespielt werden, dient das Byte hinter der letzten Note als Zeiger. Am Anfang eines Taktes wird das letzte Byte hinter der letzten Note benutzt. Folgende Werte sind einzugeben:

erstes Arpeggio = »08«
zweites Arpeggio = »10«
drittes Arpeggio = »18«
viertes Arpeggio = »20«
usw.

**<£> Sound-Options**

Die »Sound-Options« sind eine Hilfsfunktion zur bitweisen Umrechnung. Sie ermöglichen es, die Transpose- oder Soundtranspose-Funktion zu unterbinden und das Arpeggio-Play-Bit zu setzen.

- <£> führt Sie in diese Funktion
- Mit den Cursortasten wählen Sie die Bits an.
- <*> invertiert das angewählte Bit.
- Neben »sound options« wird die Ziffer angezeigt, die sich dadurch ergibt. Diese Hilfe kann man in der Track/Step-Table und in der Notendarstellung aufrufen.
- Die dargestellte Ziffer muß anschließend von Hand in die Notendarstellung übernommen werden.
- <RETURN> bringt Sie wieder zurück ins vorherige Menü.

## Abspielen und Echtzeit-Aufnahme

Um eine Melodie auf der Tastatur zu spielen, müssen Sie dem Computer verschiedene Parameter mitteilen:

**<O> / <SHIFT O> Octavetranspose (gespielte Oktave)**

Die Oktave, in der gespielt werden soll, wird rechts am Bildschirm neben »octave« angezeigt. Durch Tastendruck auf <O> zählt die Oktave hoch (C-0 bis C-7), mit <SHIFT O> nach unten.

**<K> / <SHIFT K> Keytranspose (gespielte Tonart)**

»Keytranspose« ist die Einstellung der Tonart. Sie verändern die Einstellung in Halbtonschritten durch <K> in Aufwärtsrichtung. Mit <SHIFT K> wird abwärts gezählt.

**<,> <.> </> play voice (Stimmenauswahl)**

Mit diesen Tasten wählen Sie eine oder mehrere Stimmen aus, mit der Sie auf der Tastatur spielen. Die Anzeige wird für die jeweils aktivierte Stimme (Voice) invers.

**<F1> current Step (gerade gespielter Takt)**

Ein Tastendruck auf <F1> setzt den »current step« auf den Takt, in dem der Cursor steht.

**<F3> first step (erster Takt)**

Hier wird der Anfang des Musikstücks definiert. Der Wert unter dem Cursor wird wieder übernommen.

**<F5> last step (letzter Takt)**

Bestimmt das Ende des Musikstücks. Wieder wird die Taktnummer unter dem Cursor übernommen.

**<N> new record**

Dieser Tastendruck startet das Musikstück bei »first step« und initialisiert den »SID« neu. Die Takte werden bis »last step« gespielt und ab »first step« endlos wiederholt.

**<P> Play record**

Wie »new record«, nur beginnt die Musik bei »current step«.

**<F7> play keyboard (Spielen auf der Tastatur)**

Bevor Sie <F7> drücken, sind einige Parameter zu überprüfen:

- Wählen Sie eine Stimme, die noch nicht belegt ist (siehe »play voice«).
- Die Oktave sollte einen sinnvollen Wert enthalten. Beim Ausprobieren hat sich C-4 bewährt (zur Einstellung siehe »octavetranspose«).
- Keytranspose steht normalerweise auf C.
- Sound (CLR) = »2« . Die linke Zahl bei Sound muß auf »0« stehen (bei Änderungen siehe »SOUNDOPTIONS«).
- Wenn Sie jetzt <F7> drücken, wird die Rahmenfarbe rot und Sie können über die Tastatur spielen und - falls schon Steps editiert sind - mitspielen. <CTRL> entspricht C, <1> entspricht C#. Entsprechend ist diese Reihenfolge auf der Tastatur weitergeführt.

**<R> Record (Aufnahme)**

Um eine Melodie aufzunehmen, überprüfen Sie bitte die gleichen Parameter wie unter »Play keybord«. Anschließend führen Sie folgende Schritte durch:

- Durch Drücken von <R> wählen Sie den Track an, auf dem aufgenommen werden soll. (Die Kopfzeile des angewählten Tracks verfärbt sich gelb.)
- Definieren Sie die Speicherbereiche, in die aufgenommen werden soll (siehe TRK1 bis TRK2).
- Überprüfen Sie die richtige Taktlänge (siehe »Grundeinstellungen, Byte 3).
- Quantize: Diese Einrichtung gleicht Taktfehler aus. <Q> zählt aufwärts (0 bis 3), <SHIFT Q> abwärts.

0 = kein Quantisieren.
1 = Quantisieren auf $^1/_{16}$-Noten.
2 = Quantisieren auf $^1/_8$-Noten.
3 = Quantisieren auf $^1/_4$-Noten.

- Drücken Sie <F7>. Die ausgewählten Takte werden nun endlos gespielt. Jede Toneingabe auf der Tastatur (siehe »play keyboard)« wird in Noten umgerechnet und in die markierten Takte eingetragen.

**Metronom**

Im Takt $BE80 ist nach einem Initialize ein Metronom gespeichert. Wenn Sie von Anfang an in Echtzeit einen Song eingeben möchten, ist dieser Taktgeber eine wichtige Hilfe. Setzen Sie das Metronom in eine zum Zeitpunkt der Aufnahme freie Stimme. Die »TR«- und »ST«-Spur muß dabei auf 00 stehen.

**Synchronisation**

Ein Song kann mit einem Plattenspieler oder einem Kassettendeck synchronisiert werden.

- Die Tonhöhe wird in der Soundpage unter FINE DETUNE angeglichen.
- Danach wird die Abspielgeschwindigkeit eingestellt. Dafür stehen Low- und High-Byte des CIA-Timers zur Verfügung (siehe Byte 0, Byte 1 und Byte 2 in den Grundeinstellungen).
- Mit <N> wird der Song neu gestartet.
- Wiederholen Sie diesen Vorgang so lange, bis Sie eine Übereinstimmung in der Geschwindigkeit erreicht haben.

### Lade- und Speicherfunktionen

**Save (Speichern von Datensätzen)**

Folgende drei Arten der Abspeicherung stehen zur Verfügung:

1. Soundnumber (Klänge abspeichern):

Gemeint sind die Soundparameter, die Sie mit <Z> editieren können.

- Wählen Sie durch mehrmaliges Drücken von <T>: »typ : sound y00-u00«.
- Mit den Tasten <Y> und <SHIFT Y> bestimmen Sie die Soundnummer, ab der gespeichert wird (»y00«).
- <U> und <SHIFT U> definiert die Endnummer bis zu der gespeichert wird (»x00«).
- <SHIFT S> führt Sie in den Speichermodus. Hier wandert

der Cursor auf die Zeile über den Typ. Die Eingabe erfolgt über die Tastatur.
- <RETURN> ohne Eingabe verläßt diesen Modus ohne abzuspeichern.
- Wird nur das Dollarzeichen ($) eingegeben, erhalten Sie das Directory auf dem Bildschirm.

2. Complete Song (komplettes Lied speichern)

In diesem Modus wird Ihr kompletter Sound zusammen mit einer Musikroutine auf die Diskette gespeichert:
- Setzen Sie die Parameter für »FIRST STEP» und »LAST STEP« (siehe <F1>,<F3> und <F5>).
- Kontrollieren Sie mit <N> Ihr komponiertes Musikstück. Der Song wird genauso wiedergegeben, als hätten Sie ihn absolut geladen und mit »SYS 49152« gestartet.
- Wählen Sie durch mehrmaliges Drücken von <T> den »typ : complete sound«.
- <SHIFT L> führt Sie in den Speichermodus. Hier wandert der Cursor auf die Zeile über den Typ. Die Eingabe geschieht über die Tastatur.
- Ein <RETURN> ohne Eingabe verläßt diesen Modus ohne abzuspeichern.
- Wird nur das Dollarzeichen ($) eingegeben, erhalten Sie das Directory auf dem Bildschirm.

3. Steps only (speichern ohne Abspielroutine):

Speichert den Song ohne Musikroutine. Dieser Filetyp belegt weniger Blocks auf der Diskette und bietet sich an, wenn Sie die Arbeit an einem Song später fortsetzen wollen.
- Setzen Sie die Parameter für »FIRST STEP» und »LAST STEP« (siehe <F1>,<F3> und <F5>).
- Wählen Sie durch mehrmaliges Drücken von <T> den »typ : complete sound«.
- Führen Sie die weiteren Schritte ab <SHIFT S> unter »complete song« durch.

**Load (Laden eines Datensatzes)**

Laden eines Songs:
- Wählen Sie mit <T> den Filetyp aus.
- Beim Laden eines Songs ist nur »complete Song« und »steps only« zulässig.
- <SHIFT L> führt in die Eingabezeile. Tippen Sie den Filenamen ein oder brechen Sie (ohne Eingaben) mit <RETURN> ab.
- Wird nur das Dollarzeichen ($) eingegeben, erhalten Sie das Directory auf dem Bildschirm.

Laden einer oder mehrerer Soundnummern:
- Stellen Sie mit <T> den Filetyp auf »typ : sound y00 u00«
- Durch mehrmaliges Drücken von <Y> und <SHIFT Y> bestimmen Sie die Soundnummer, ab der geladen wird (»y00«).
- <SHIFT L> führt in die Eingabezeile. Tippen Sie den Filenamen ein oder brechen Sie (ohne Eingaben) mit <RETURN> ab.
- Wird nur das Dollarzeichen ($) eingegeben, erhalten Sie das Directory auf dem Bildschirm.

**Directory (Ausgabe des Disketteninhalts)**

Das Inhaltsverzeichnis der Diskette wird folgendermaßen sichtbar gemacht:
- <SHIFT L> oder <SHIFT S> führt zur Eingabezeile.
- Die $-Taste <SHIFT 4> erzeugt das Inhaltsverzeichnis.
- Drücken einer beliebigen Taste schaltet zurück auf den Track/Step-Bereich.
- Hier dürfen keine Eingaben gemacht werden.
- <RETURN> bringt Sie zum Ausgangspunkt zurück.

## Das Musikfile des »Soundmonitor«

Der Vorteil gegenüber den meisten anderen Soundeditoren liegt in der völlig unabhängigen Abspielroutine. Sie funktioniert auch ohne den Soundmonitor. Beim Speichern als »complete song« wird diese Routine an Ihre Komposition gehängt. Dieses Musikprogramm muß unmittelbar geladen werden (mit »8,1«) und belegt keinen Platz im Basic-Speicher. Haben Sie Ihr Werk z.B. unter »MUSIK 1« auf Diskette gespeichert, sieht der Ladevorgang folgendermaßen aus:

```
LOAD"MUSIK 1",8,1
```

und <RETURN>. Anschließend geben Sie NEW und <RETURN> ein und initialisieren die Routine mit SYS 49152. Im Hintergrund läuft die Musik ohne Ihr Zutun ab, während Ihnen der C64 zur Verfügung steht.

### Hilfsprogramme zum Soundmaster

**Musik-Switch (Hilfsroutine zum Ein- oder Ausschalten)**

Auf der beiliegenden Diskette finden Sie das Programm »MUSIK-SWITCH«. Damit ist es möglich, den Lauf des unter dem Soundmonitor erstellten Musikstückes durch Angabe von SYS828,x(,y) und entsprechenden Steuerparametern zu beeinflussen. Sie laden dieses Hilfsprogramm mit:

```
LOAD"MUSIK-SWITCH",8,1
```

**Dieses Programm funktioniert nur, wenn sich ein Musikfile im Speicher befindet!**

Beschreibung der Funktionen:

| | |
|---|---|
| SYS828,0 | hält das Musikstück sofort an |
| SYS828,1 | setzt das Musikstück an der Stelle fort, an der es mit SYS828,0 angehalten wurde |
| SYS828,2 | startet das Musikstück von Anfang an |
| SYS828,x,y | spielt die Steps von Step x zu Step y. Mit diesem Befehl wäre es möglich, mehrere kleine Musikstücke innerhalb einer Musikroutine zusammenzustellen und einzeln aufzurufen. |

**Sound-Cruncher (Datenkompressor für Ihre Musik)**

Die Länge der unter dem Soundmonitor erstellten Daten beträgt 45 Blocks. Dabei ist es gleichgültig, wie lang die Musik tatsächlich ist. Für alle, die keinen Floppy-Speeder zur Verfügung haben, nimmt das Laden doch einige Zeit in Anspruch. Außerdem werden die Musik-Files absolut, also mit dem Zusatz »,8,1«, geladen und mit SYS49152 gestartet.

Der Sound-Cruncher beseitigt diese Probleme. Er komprimiert Ihre Musikstücke mit einem speziell auf deren Format zugeschnittenen Algorithmus. Das Ergebnis ist beachtlich: Von 45 Blöcken des Musikstücks »Axel F.« bleiben nach dem Packen lediglich 18 Blocks übrig.

Zum Komprimieren eines Ihrer Musikstücke laden Sie den Sound-Cruncher von der beiliegenden Diskette mit:

```
LOAD"SOUND-CRUNCHER",8
```

und starten es mit RUN. Legen Sie nun die Diskette in Ihr Laufwerk, die das zu packende Musikstück enthält. Diesen Namen müssen Sie als erstes eingeben.

Nach der Bestätigung mit <RETURN> lädt der Sound-Cruncher dieses Musikstück und komprimiert es. Anschließend erfolgt die Eingabe des Namens, den das fertig gepackte Musikprogramm auf Diskette erhalten soll. Setzen Sie beispielsweise ein »P/« vor den Namen. So lassen sich komprimierte von normalen Musik-Files unterscheiden.

Achtung: Gepackte Musikstücke lassen sich mit dem Soundmonitor nicht laden, starten oder editieren.

Wenn Sie ein gepacktes Musikstück starten möchten, laden Sie es wie ein normales Basic-Programm mit »LOAD "NAME",8« und starten es mit RUN.

**Beispieldemos auf Diskette**

Auf der beiliegenden Diskette finden Sie zwei Musikfiles, die mit dem Soundmonitor erstellt wurden: »AXEL F.« und »DANCE AT NIGHT«. (Chris Hülsbeck/gr)




# Rauhe Sitten auf glattem Eis

**Eishockey – das schnellste Mannschaftsspiel der Welt. In manchen Ländern der Erde ist es beliebter als Fußball. Trotz seiner Kürze ist »Bully« eine unwahrscheinlich rasante Simulation dieses Kampfspiels für harte Männer.**

Wenn die Stimmung auf dem Höhepunkt angelangt ist, und die Joysticks knirschen, als würden sie im nächsten Augenblick ihren Geist aufgeben, dann kann man sicher sein: Es wird »Bully« gespielt.

Laden Sie das Programm mit

```
LOAD "BULLY",8
```

und starten Sie es mit RUN.

Eine Eisfläche mit vier Spielern erscheint auf dem Bildschirm, aus der Vogelperspektive betrachtet (Bild 1). Die beiden Spielfiguren in den roten Dressen (Torwart und Feldspieler) steuern Sie mit dem Joystick in Port 1, die grüngekleideten Gegner mit dem Steuerknüppel in Port 2.

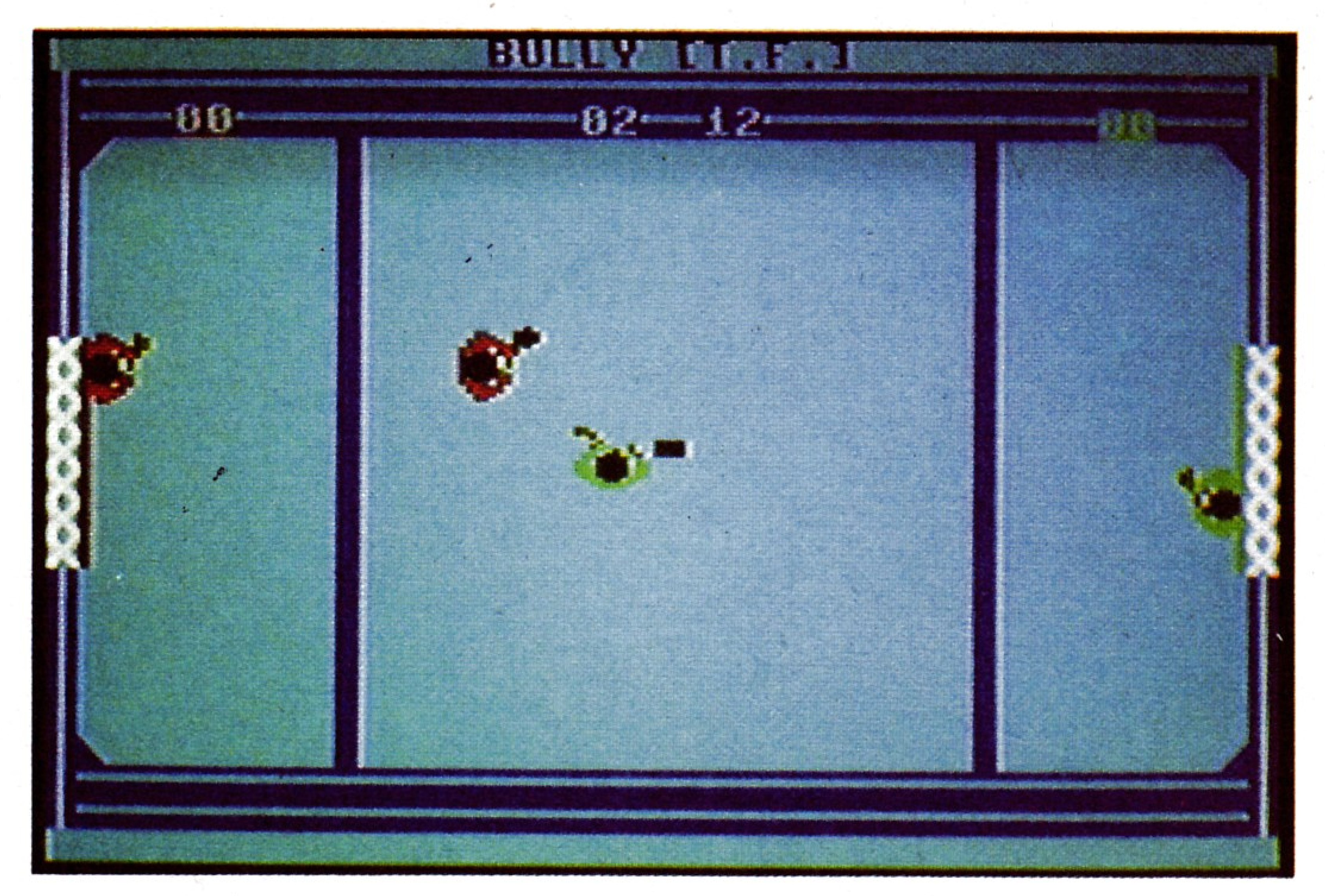



**Bild 1. Schnelle Reaktionen und stabile Joysticks – das sind gute Voraussetzungen für »Bully«**

Um das Spiel zu starten, müssen beide Spieler gleichzeitig den Feuerknopf ihres Joysticks drücken. Die schwarze Scheibe, der Puck, wird dadurch ins Spielfeld befördert. Das Spiel beginnt.

Die Angriffsspieler werden mit Joystickbewegungen über die Eisfläche gejagt. Man sollte sich daran gewöhnen, daß der Stürmer manchmal an der Scheibe vorbeirutscht. Beschleunigung und Bremskraft spielen wie beim »echten« Eishockeyspiel eine wichtige Rolle.

Haben Sie es geschafft, den Puck zu ergattern, klebt er am eigenen Schläger. Sie können in Richtung gegnerisches Tor davonziehen und aus einer günstigen Position heraus mittels Feuerknopf einen Schuß abfeuern. Geben Sie acht, daß Ihnen der andere Stürmer nicht zu nahe kommt. Er könnte Ihnen sonst die Scheibe wieder abknöpfen.

Echte Eishockey-»Cracks« nutzen für Dribblings und Schüsse auf das Tor intensiv das Spiel mit der Bande. Vorsicht: Nicht selten geht der Schuß im wahrsten Sinne des Wortes nach hinten los und überwindet den eigenen Torhüter. Wenn Sie den Feldspieler während des Spielverlaufs vertikal bewegen, macht der Tormann diese Bewegung mit.

Zu Beginn des Spiels ist es nicht einfach, die eigenen Spielfiguren richtig zu koordinieren und auf den Bewegungsablauf des Pucks einzustellen. Nach einigen Trainingsspielen haben Sie die Eishockeyspieler mit dem Joystick im Griff.

Gewonnen hat die Mannschaft, die in drei Minuten die meisten Treffer erzielt. Sind Sie vom Spiel so begeistert, daß Ihnen diese Zeit zu kurz ist? Unmittelbar nach dem Laden kann mit

```
POKE 2193, Zeit
```

die Spieldauer nach Wunsch verändert werden. Für »Zeit« müssen Sie einen Wert zwischen 0 und 255 einsetzen. Die echte Spielzeit errechnet sich, indem Sie die gewählte Zahl (z.B. 3) mit der im Programm voreingestellten Zeit (100 s) multiplizieren. In unserem Beispiel beträgt demnach die Spieldauer 300 s (5 min).

## Programmtechnische Hinweise

»Bully« wurde in Maschinensprache programmiert und belegt den Speicherbereich von $0801 (2049) bis $0FE3 (4067). Es besitzt eine Basiczeile mit dem Startbefehl (SYS 2061). Nach dem Spielstart wird die Tastenkombination <RUN/STOP RESTORE> unterbunden. Abbrechen können Sie lediglich mit einem RESET oder durch Aus- und Einschalten des C 64. Die Spielgrafik- und Spritedaten befinden sich als Tabellen am Programmende. Die Interruptvektoren, die normalerweise auf die Adresse $EA31 (59953) zeigen, sind auf die Hauptroutinen des Programms ab Speicherstelle $0C61 (3169) gerichtet. Dies gewährleistet den rasanten Spielablauf.

Wir wünschen Ihnen bei »Bully« gut geschliffene Schlittschuhkufen, strapazierfähige Joysticks und viele Tore.

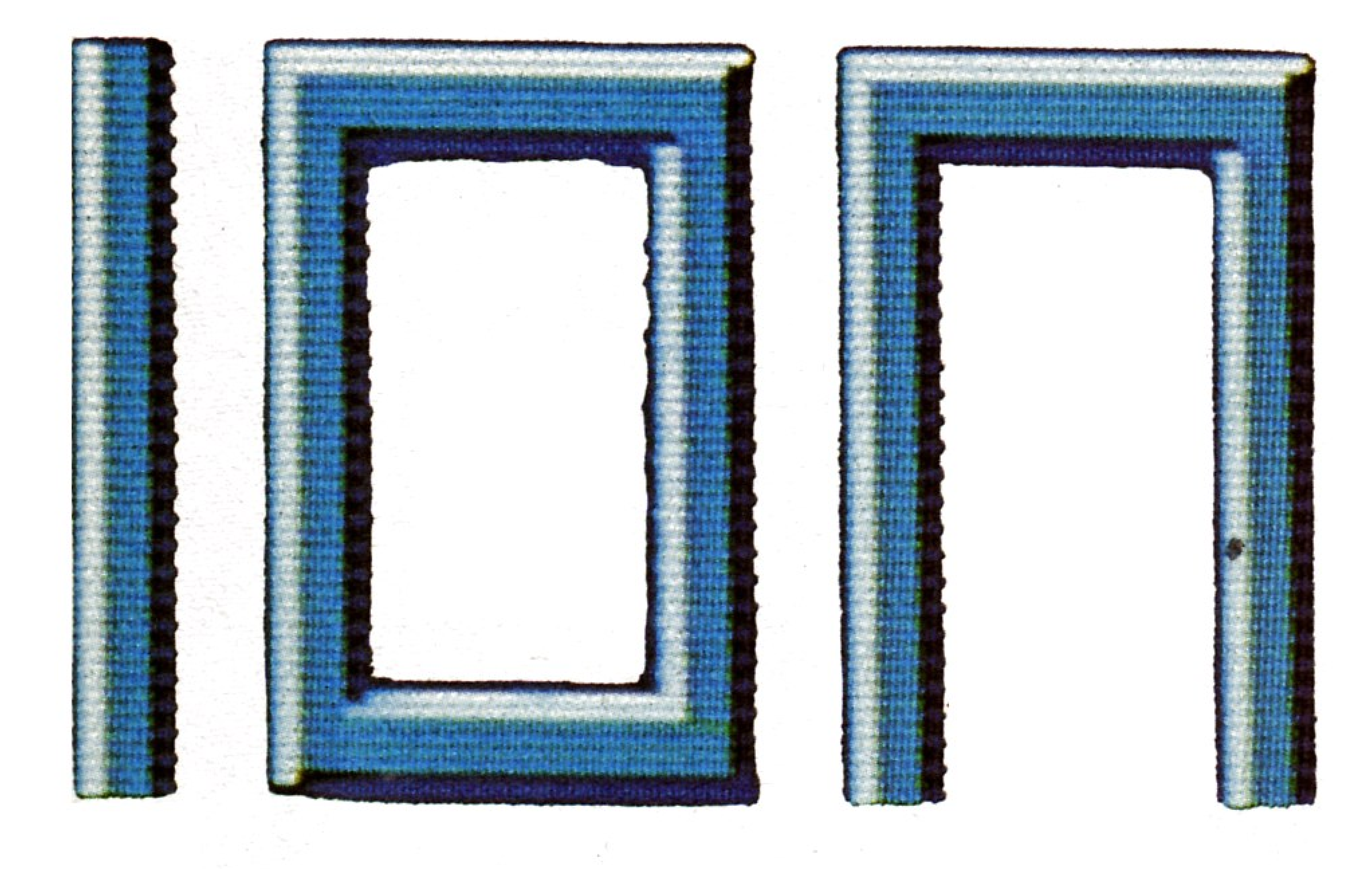
(Thorsten Feiweier/bl)

## Kurzinfo: Bully

**Programmart:** Eishockey-Simulation für zwei Spieler
**Laden:** LOAD "BULLY",8
**Start:** Nach dem Laden RUN eingeben
**Spielziel:** Sieger ist, wer innerhalb des Zeitlimits die meisten Treffer erzielt
**Steuerung:** Zwei Joysticks (Port 1 und 2) dirigieren die Bewegungen der Spieler
**Besonderheiten:** Spielbeginn durch gleichzeitiges Drücken der Feuerknöpfe beider Joysticks. Der Torhüter bewegt sich nur in vertikaler Richtung. Ein Schuß wird mit dem Feuerknopf ausgelöst.
**Benötigte Blocks:** 8 Blocks
**Programmautor:** Thorsten Feiweier




**Neben dem legendären »Ping Pong« war »Breakout« eines der ersten Computerspiele, die es überhaupt gab. Mit »Crillion« stellen wir Ihnen eine faszinierende Variante dieses Spiels vor.**




Erinnern Sie sich an das Computerspiel »Breakout«? Trotz einfachen Spielprinzips und »spartanischer« Grafik bestach es durch hohe Spielmotivation. Mit einem Schläger, der am unteren Bildschirmrand hin- und herbewegt werden konnte, mußte ein Ball auf eine steinerne Wand geschossen werden. Jeder getroffene Stein erhöhte den Highscore.

Die Idee dieses Spiele-Evergreens ist wieder »in«, wobei die fantastischen Grafik- und Soundeigenschaften des C64 voll genutzt werden. Sie statten dieses eher karge Spiel mit ausgezeichneten Variationen (Levels) aus. »Crillion« reiht sich würdig in die Gruppe der Geschicklichkeitsspiele ein. Es handelt sich um eine völlig neue Variante der Spielidee von »Breakout«.

Laden Sie das Programm mit folgender Anweisung:

```
LOAD "CRILLION",8
```

Gestartet wird mit RUN.

Bei jedem Spielstart lädt »Crillion« automatisch die aktuelle Highscore-Liste »Top 8« von der beiliegenden Diskette. Der Joystick muß sich in Port 1 befinden. Damit kann der Spielball lediglich nach rechts oder links bewegt werden. Nach oben oder unten läßt sich die selbständige Ballbewegung nicht beeinflussen.

Im Gegensatz zu anderen Programmen mit derselben Spielidee besitzt »Crillion« keinen Schläger, der den Ball in bestimmte Richtungen lenkt – man steuert den Spielball selbst. Während er unbehelligt von den Bemühungen des Spielers auf und ab hüpft, kann er nach links oder rechts dirigiert werden. Dies ist ein vollkommen neues Spielgefühl.

## Mit »Haken und Ösen«

Das Spielfeld ist rundum abgeschlossen. Dadurch kann der Ball nicht verlorengehen oder hinter der Grafik verschwinden. Genügend andere Fallen und Hindernisse lauern auf Sie und erschweren den Abschuß aller Spielsteine.

Tödliche Felder, die Sie auf keinen Fall berühren dürfen, vernichten den Ball (er löst sich in eine Staubwolke auf). Diese Felder sind bezeichnenderweise durch einen Totenkopf gekennzeichnet. Dieser zwinkert dem Spieler bei der Zerstörung des Balles hämisch zu.

Feste Mauern blockieren Eingänge. Verschiebbare Steine, die wie Disketten aussehen, verwehren Ihnen den Weg. Wenn Sie diese Blöcke unachtsam verschieben, versperren Sie sich den Druchgang zu manchem Spielstein. Er läßt sich in so einem Fall nicht mehr bewegen, das aktuelle Level kann dadurch nicht beendet werden. Dies geschieht nur durch das Beseitigen aller Steine, die sich auf dem Spielfeld befinden. Abhilfe schafft die »Selbstzerstörung« durch Lenken des Balles auf ein Todesfeld oder der Druck auf die Taste <RESTORE>.

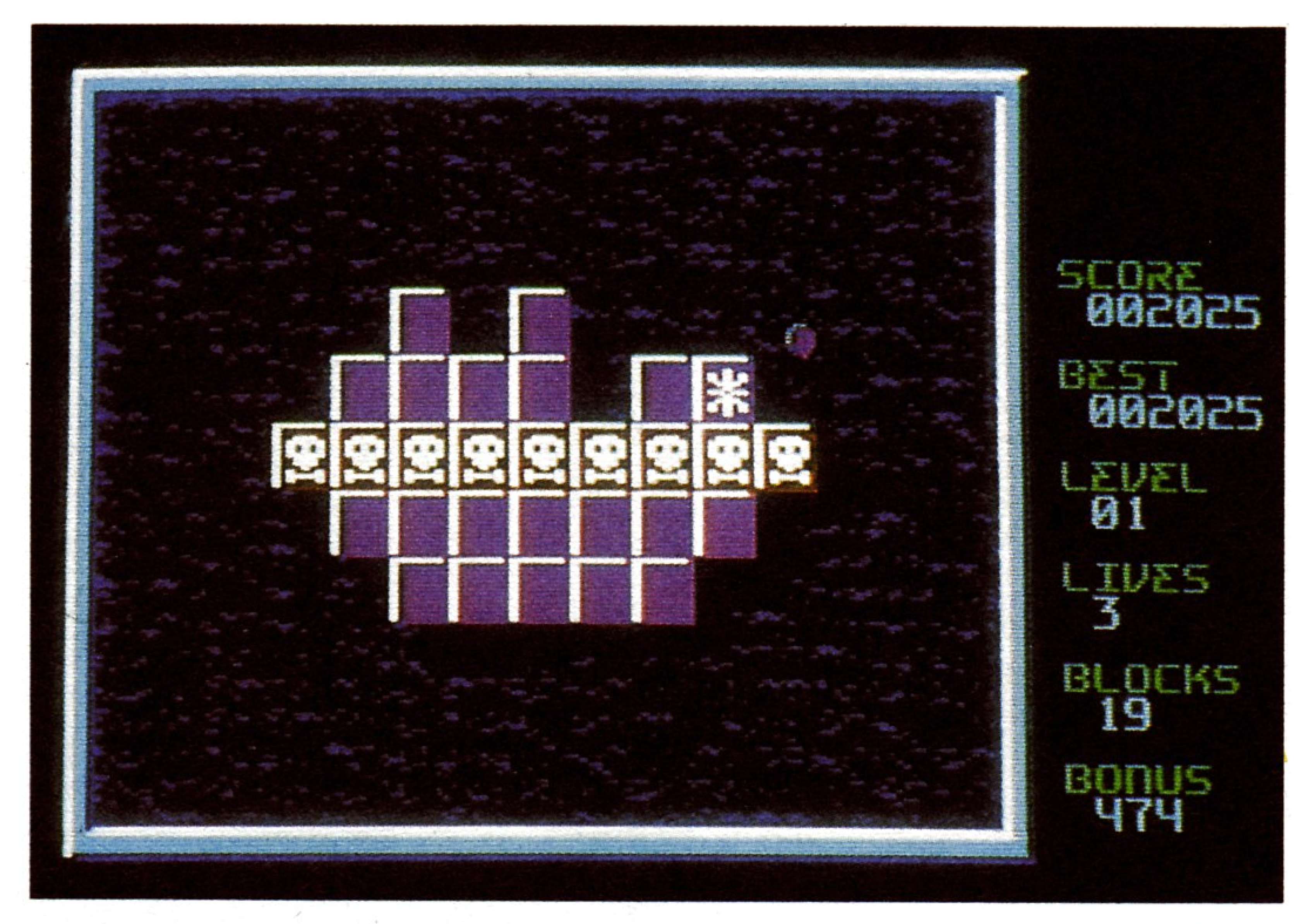



**Bild 1. Der erste Level von Crillion. Er ist noch recht einfach zu lösen.**

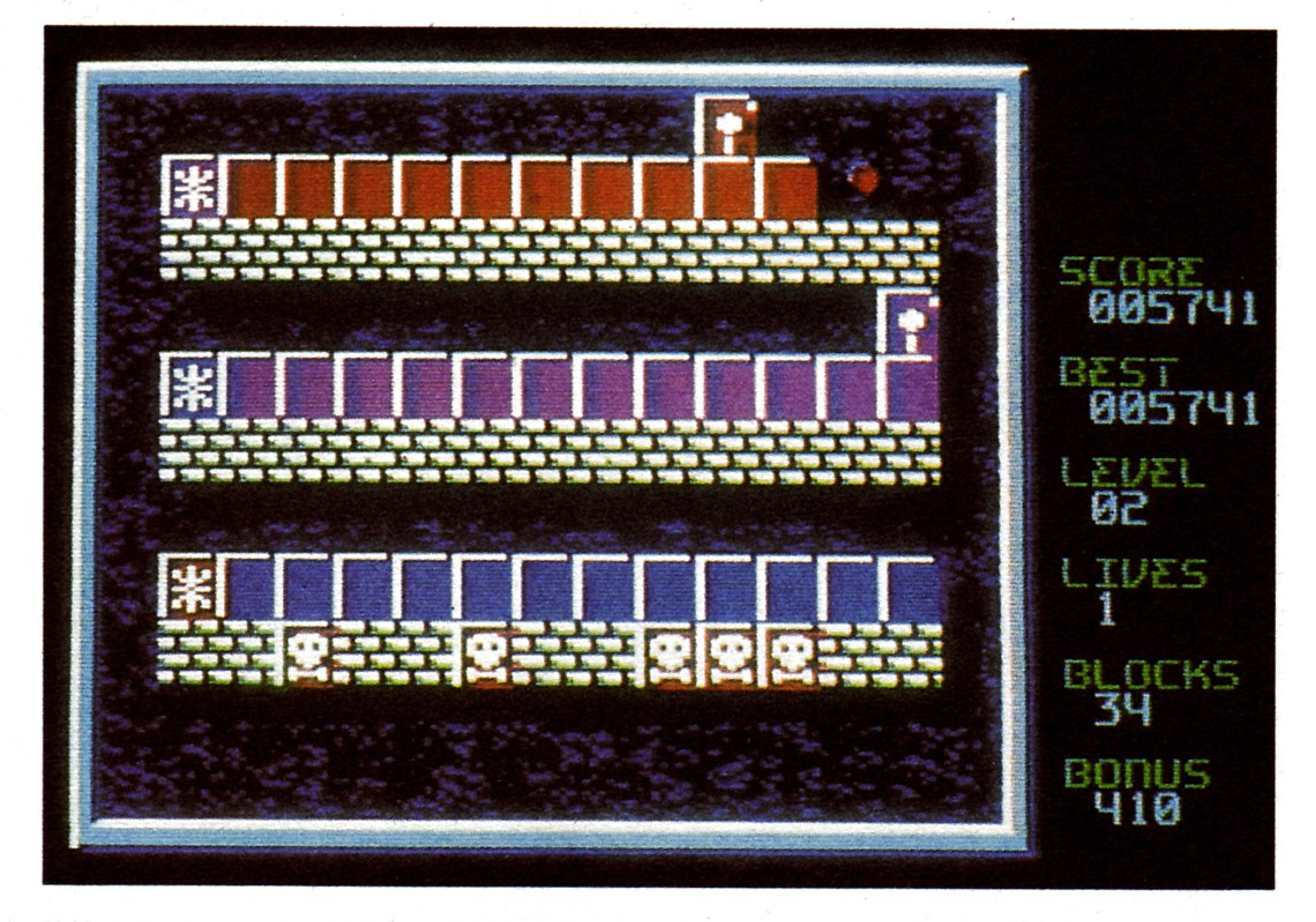



**Bild 2. Level 2 läßt sich mit einem Trick lösen. Verschieben Sie die Diskettensymbole von der rechten Seite.**

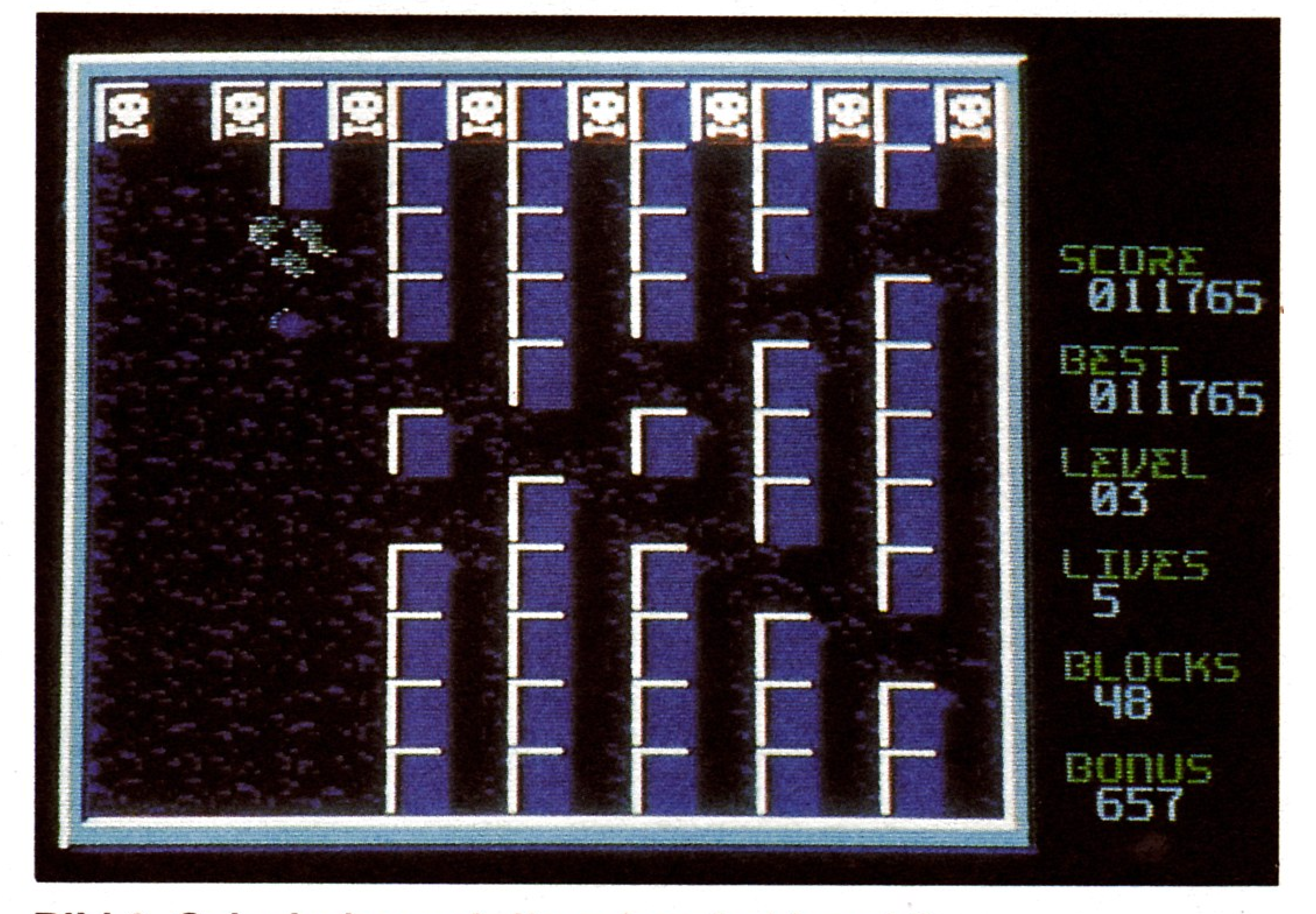



**Bild 3. Schwieriger wird's schon bei Level 3**

Ein Spielfeld (Level) setzt sich aus verschiedenartigen Steinen zusammen. Sie können erst dann abgeschossen werden, wenn der Ball die gleiche Farbe angenommen hat. Andernfalls prallt er wirkungslos vom Steinblock ab. Um die Farbe zu wechseln, muß der Spielball ein dafür vorgesehenes Feld berühren. Dieses ist mit einem Sternchen-Muster gekennzeichnet. Oft kann eine Spielstufe nur durch überlegtes und geschicktes Wechseln der Ballfarbe und Abschießen gewisser Steine gelöst werden.

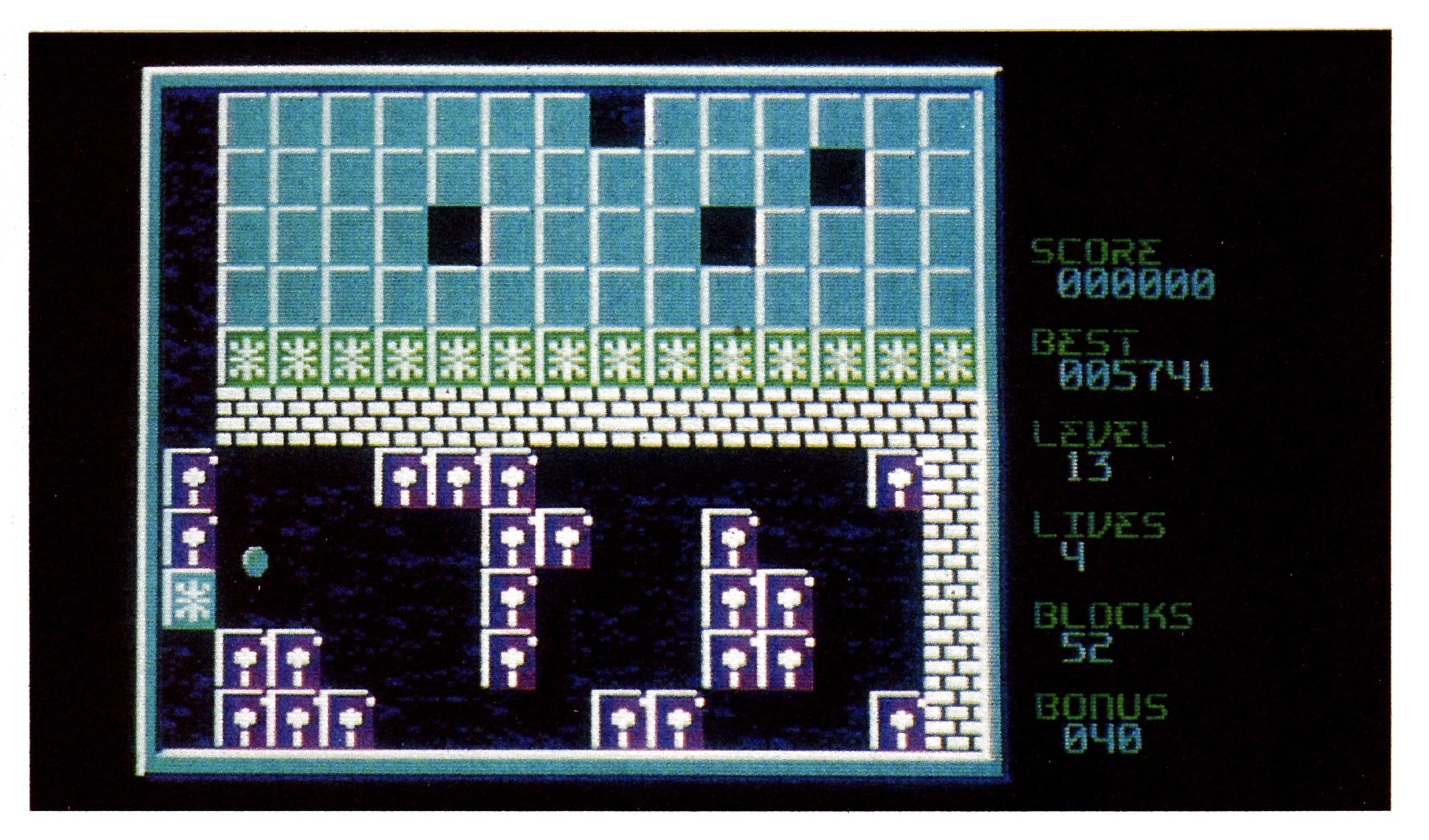
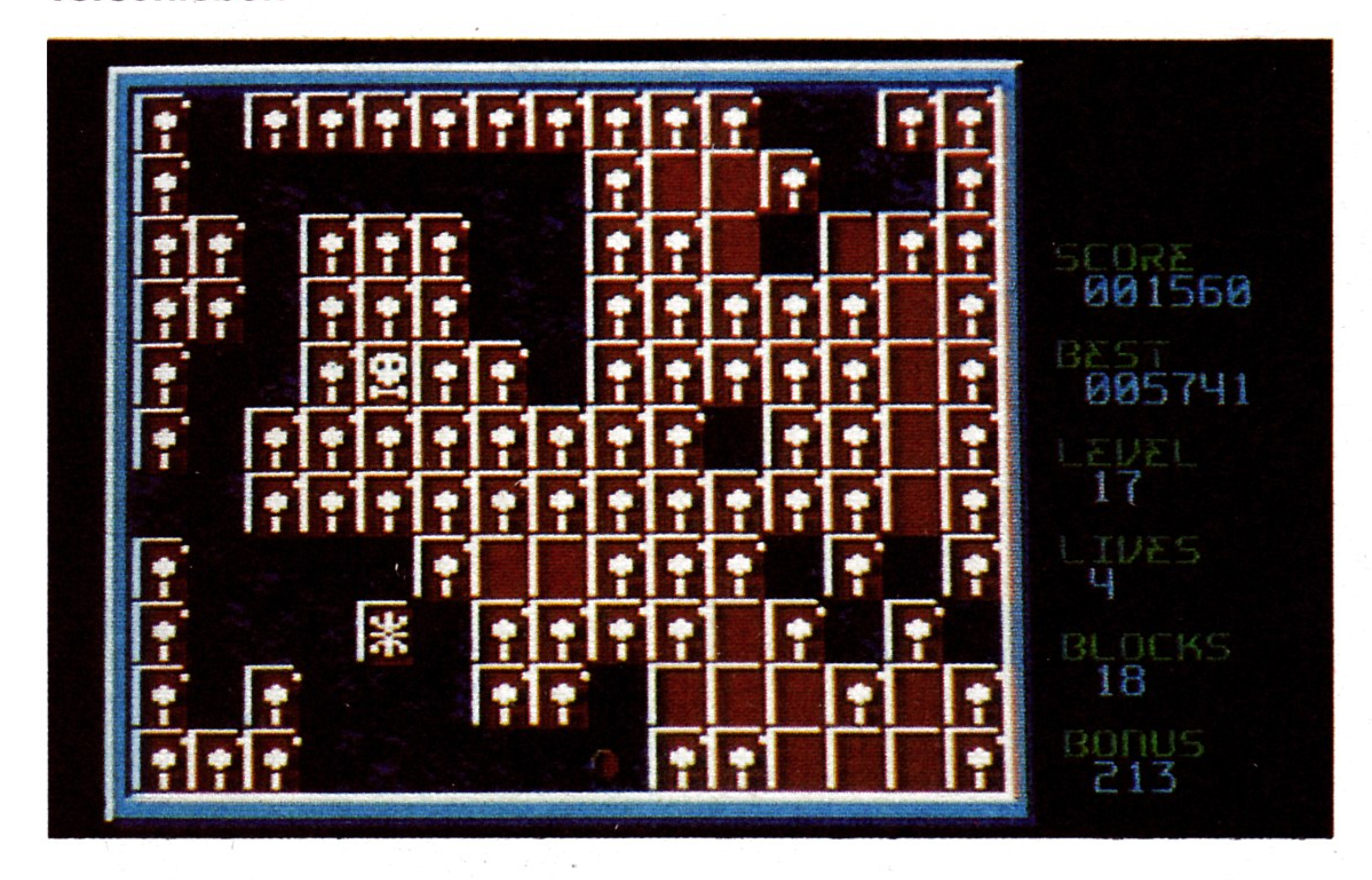



**Bild 4.** ▶ **In Level 13 hat sich der Spieler in eine auswegslose Situation manövriert**

▼ **Bild 5. In Level 17 heißt es, jede Menge Steine zu verschieben**




Je nach Spielstufe erhalten Sie für jeden abgeschossenen Stein unterschiedliche Punktezahlen. Wer beim »Aufräumen«

## Die Highscore-Liste

des Levels das gesetzte Zeitlimit nicht überschreitet, erhält zusätzliche Bonuspunkte.

Verliert man einen Ball (durch ein Todesfeld oder Selbstzerstörung), muß die aktuelle Spielstufe erneut durchgespielt werden. Haben Sie alle Steine eines Levels zerstört, kommen Sie automatisch in das nächste Spielfeld. Sie dürfen sicher sein, daß dieses Level schwieriger zu bewältigen ist als das vorhergehende. Das Spiel besitzt 25 Spielfelder. Fünf Beispiele zeigen Ihnen die Bilder 1 bis 5. Ist die letzte Spielstufe (Nummer 25) erreicht und erfolgreich überstanden, endet das Spiel. Endlich ist es geschafft!

Falls Ihre erzielte Punktezahl hoch ist, können Sie Ihren Namen in die Highscore-Liste »Top 8« eintragen, die auf Diskette zurückgeschrieben wird.

### Kurzinfo: Crillion

**Programmart:** Geschicklichkeitsspiel mit 25 Levels
**Spielziel:** Alle einfachen Steine eines Levels müssen mit einem beweglichen Ball abgeschossen werden. Spielprinzip ähnelt »Breakout«.
**Laden:** LOAD "CRILLION",8
**Starten:** Nach dem Laden RUN eingeben
**Steuerung:** Joystick Port 1
**Besonderheiten:** Highscore-Liste wird auf Diskette gespeichert und bei Programmstart geladen
**Benötigte Blocks:** 45 Blocks
**Programmautor:** Oliver Kirwa

## Hinweise für Assembler-Programmierer

»Crillion« ist in Maschinensprache programmiert und belegt den Basic-Speicher von $0801 (2049) bis $330D (13069). Das Programm benutzt zur grafischen Darstellung der Levels den Extended-Color-Modus sowie einen veränderten Zeichensatz. Der Bildschirm wird nach $CC00 (52224) verlegt. In Adresse 648 steht die Zahl »204«.

Wie bei allen rasanten Action-Games mußten hier ebenfalls die Betriebssystemvektoren für den Interrupt »verbogen« werden. Der IRQ-Vektor $0314/$0315 (Normalinhalt: $EA31) zeigt auf eine Routine im Programm ab Adresse $1EC7 (7879), die die Speicherstellen $D019 (53273) und $DC0D (56333) ausliest. Die beiden Zeiger des NMI-Vektors (Normalinhalt: $FE47) sind auf die Adresse $FEC1 (65217) im Kernel des Betriebssystems gerichtet. Diese Umstellung bewirkt einen Rücksprung zum aktuellen Level (z.B. nach Druck auf die Selbstzerstörungstaste <RESTORE>), ohne das Spiel abzubrechen oder Farben und Zeichensatz zu ändern. Der eigentliche Spielstart liegt bei $0E16 (3606).

Das Spiel besitzt eine Highscore-Liste mit acht möglichen Einträgen. Der Filename »Top 8« ist ab Adresse $226C (8812) abgelegt und besteht aus fünf Zeichen. Um die Liste nach jedem Spiel aktualisieren und auf Diskette speichern zu können, wird die REPLACE-Funktion des C64 benutzt: ein vorangestelltes »At Sign«-Zeichen (Klammeraffe) mit anschließendem Doppelpunkt.

Die Routinen zum Laden und Speichern der Liste der »Besten« finden Sie ab den Adressen $1F64 bzw. $2174. Die Daten der einzelnen Levels stehen im Bereich ab $2271 bis zum Programmende.

Worauf warten wir eigentlich noch? Joystick eingestöpselt, und los geht's! 25 Spielstufen sind eine ganze Menge, aber als Belohnung winkt der 1. Platz im Highscore. Viel Spaß!

(Oliver Kirwa/bl)

Markt&Technik

# 64'er SOFTWARE EXTRA

# SOFTWARE DER EXTRAKLASSE

## Grafik



**64'er Extra Nr. 1:**
**The Best of Grafik**
Giga-CAD. Hi-Eddi. Title-Wizzard, Filmkonverter.
Bestell-Nr. 38701
**DM 49,90***
(sFr 44,90*/öS 499,–*)



**64'er Extra Nr. 2:**
**The Best of Grafik**
Tolle Grafik-Erweiterungen.
Bestell-Nr. 38702
**DM 39,90***
(sFr 34,90*/öS 399,–*)



**64'er Extra Nr. 3:**
**The Best of Grafik**
Erweiterungen für Grafik und Spiele. 3-D-Trickfilm. Apfelmännchen. Super-Hardcopies.
Bestell-Nr. 38703
**DM 39,90***
(sFr 34,90*/öS 399,–*)



**64'er Extra Nr. 17:**
**Aus der Wunderwelt der Grafik**
EGA: Sramycs Sprite-Graphics: 51 neue Basic-Befehle.
Bestell-Nr. 38757
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**64'er Extra Nr. 18:**
**Das Beste aus der Welt der Grafik**
Ped. Dreher. Perspektiven: Grafiken mit räumlicher Tiefe versehen.
Bestell-Nr. 38758
**DM 49,–***
(sFr 45,–*/öS 490,–*)

## Spiele



**64'er Extra Nr. 4:**
**Abenteuer-Spiele**
Robox: Adventure. Scotland Yard: Kriminaladventure.
Bestell-Nr. 38704
**DM 29,90***
(sFr 24,90*/öS 299,–*)



**64'er Extra Nr. 15:**
**Abenteuer-Spiele**
»Der verlassene Planet« und »Mission«. Befreien Sie die Erde von den Dämonen.
Bestell-Nr. 38730
**DM 39,–***
(sFr 35,–*/öS 390,–*)

## Anwendungen und Utilities



**64'er Extra Nr. 9:**
**Abenteuer-Spiele**
Wanderung/Sein letzter Trick: 2 Text-Adventures garantieren spannende Unterhaltung.
Bestell-Nr. 38715
**DM 39,–***
(sFr 35,–*/öS 390,–*)



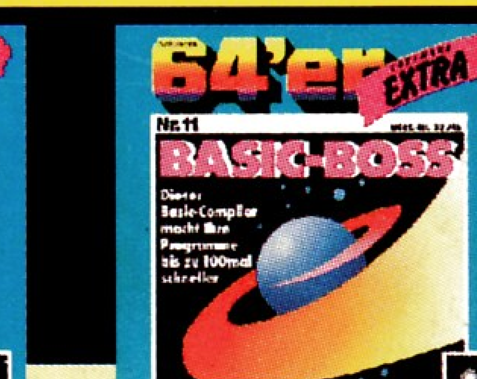
**64'er Extra Nr. 10:**
**Spiele**
Rebound: Duell – eine Arena im Jahre 2574. Palobs – ganz entfernt von Dame.
Bestell-Nr. 38742
**DM 39,–***
(sFr 34,–*/öS 390,–*)



**64'er Extra Nr. 6:**
**The Best of Floppy-Tools**
Programme für den täglichen Einsatz Ihrer Disketten-station.
Bestell-Nr. 38707
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**64'er Extra Nr. 7:**
**Programmier-Utilities**
Eine Sammlung leistungsfähiger Basic-Befehlserweiterungen.
Bestell-Nr. 38716
**DM 39,–***
(sFr 35,–*/öS 390,–*)



**64'er Extra Nr. 11:**
**Basic-Boss**
Dieser Basic-Compiler macht Ihre Programme bis zu 100mal schneller.
Bestell-Nr. 38745
**DM 49,–***
(sFr 45,–*/öS 490,–*)



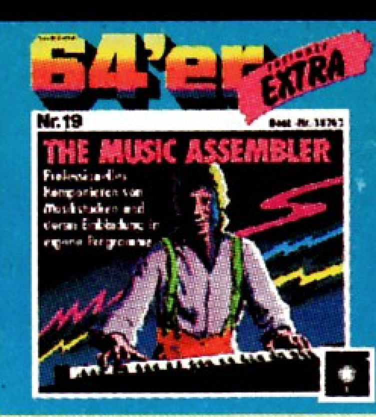
**64'er Extra Nr. 12:**
**GSF-System**
Ein leistungsstarkes Programmiersystem zum Schreiben von Programmen im GEM-Look.
Bestell-Nr. 38731
**DM 49,–***
(sFr 45,–*/öS 490,–*)



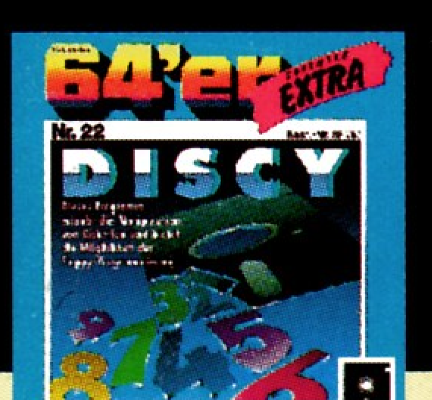
**64'er Extra Nr. 13:**
**The Best of Anwendungen**
Soundmonitor, Mony 64, Proterm V6. Giga-ASS.
Bestell-Nr. 38717
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**64'er Extra Nr. 14:**
**The Best of Anwendungen**
Master-Tool. Smon und Promon. Mailbox. Datec.
Bestell-Nr. 38720
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**64'er Extra Nr. 19:**
**The Music-Assembler**
Erstellen Sie auf einfachste Weise eigene Musikstücke!
Bestell-Nr. 38763
**DM 49,–***
(sFr 45,–*/öS 490,–*)

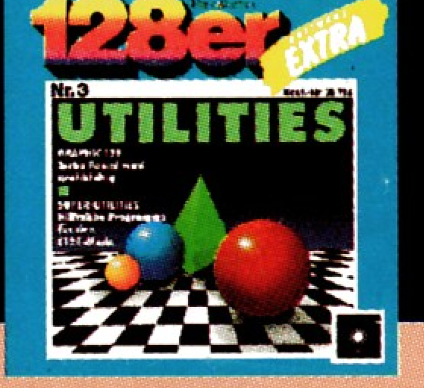


**64'er Extra Nr. 22:**
**Disky**
Manipulation von Disketten. Floppy-Programmierung.
Bestell-Nr. 38767
**DM 49,–***
(sFr 45,–*/öS 490,–*)

## C 128 und Plus/4



**128er Extra Nr. 1:**
**The Best of 128er**
Mastertext 128. Color Pack 1. Double-Ass. Utilities.
Bestell-Nr. 38712
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**128er Extra Nr. 3:**
**Utilities**
Graphic 128: Turbo Pascal wird grafikfähig. Super-Utilities: Hilfreiche Programme.
Bestell-Nr. 38713
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**128er Extra Nr. 2:**
**Paint R.O.I.A.L.**
Ein Malprogramm, das die höchste Auflösung Ihres C128 verwendet.
Bestell-Nr. 38736
**DM 49,–***
(sFr 45,–*/öS 490,–*)



**64'er Extra Nr. 8:**
**MasterBase Plus/4**
Eine semiprofessionelle Dateiverwaltung mit vielen Leistungsmerkmalen.
Bestell-Nr. 38719
**DM 49,–***
(sFr 45,–*/öS 490,–*)

*Unverbindliche Preisempfehlung

## INFO-COUPON

Bitte senden Sie mir Ihr Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software.

Name ____________________

Straße ____________________

PLZ/Ort ____________________

Bitte ausschneiden und einsenden an: Markt&Technik Verlag AG, Buch- und Software-Verlag, Hans-Pinsel-Straße 2, 8013 Haar bei München 64 SH 53




Zeitschriften · Bücher
Software · Schulung

Markt&Technik-Bücher und -Software erhalten Sie bei Ihrem Buch- oder Computerfachhändler

232/909